Analysis of Flow in Stented Aneurysm

Gabriela M. Ong
MSc Applied Mathematics, EEMCS

University of Twente

Supervisors
Prof. Dr. Ir. Bernard Geurts
Dr. Julia Mikhal

August 2018

Abstract

Among the challenges in the simulation of haemodynamics in the human brain is the problem of connecting the raw data from medical imagery to a quantitative, patient-specific prediction of the flow in order to provide information assisting practitioners when deciding on a treatment. In this study a sequence of steps is presented to facilitate such connection, focusing on an endovascular treatment known as flow-diverter stenting. Starting from DICOM data, the format changes and preparations based on Horos and Blender are described, as pre-cursors to an immersed boundary (IB) representation of the geometry. On the IB representation, a fluid mechanics analysis within the CFD toolbox OpenFOAM is done, providing both qualitative and quantitative descriptions of the flow. This sequence of steps is discussed and illustrated for flow in a model geometry with the typology of a side-wall aneurysm, under steady-state and pulsating flow conditions. The flow diverting effect of two different stents is presented, showing a significant blocking of flow from entering the aneurysm sac after stent placement. Under suitable spatial resolution conditions accurate results are obtained and we observe that both a fine-mesh flow diverter and an eCLIPS device perform virtually identically in terms of almost completely halting the flow in the aneurysm sac. A slight preference for an eCLIPS can be motivated, showing a more uniform flow blocking also in the immediate vicinity of the device, combined with more flexibility of merging it with the diseased vessel.

Contents

Chapter 1

Introduction

In treating a medical condition called aneurysms, endovascular treatment has been preferred over open surgery, due to its less invasive nature. A type of this treatment is the so-called flow-diverting stenting, in which a stent is placed in the blood vessel to divert the flow away from the aneurysm sac, thus reducing the risk of rupture and encouraging the repair of the vessel. The challenge of this method is on choosing the correct geometry and placement of the stent in order to create the most effective flow-diversion in the aneurysm of a specific patient. Moreover, this choice will affect the healing process following the stent placement.

With the advancement of medical imaging which produces high-precision record of a patient's anatomy, unruptured aneurysms can now be detected early and intervention measures can then be planned. Additionally, the high-precision medical imaging record of a patient's diseased vessel, combined with tools to process those images also allows for generating high-fidelity geometry representations. In turn, this has stimulated studies aiming to present mathematical models predicting the flow for untreated aneurysms [15] or for a variety of possible treatments, including virtual stenting [24] among others. The result of such studies can then go back to the practitioners, adding information they could consider in their decision making, such as the comparison of treatment options ahead of surgery and the optimization of stents for particular situations.

In this study, we consider the overall approach linking on the one hand raw medical imagery of a patient to, on the other hand, the final bottom-line presentation of the outcome of a simulation study. For that, a sequence of steps is discussed, with which an analysis of the flow is generated for stented intracranial aneurysms. The initial medical images in DICOM format are converted in the software Horos and then processed through the 3D-creation suite Blender, allowing for the application of the immersed boundary representation of the geometry. On this representation, a fluid mechanics analysis within the CFD toolbox OpenFOAM is done, providing qualitative and quantitative description of the flow. The sketch of this workflow is depicted in fig. 1.1 (see also [17]).

The use of Blender in this study enables us to virtually place stents inside the aneurysm

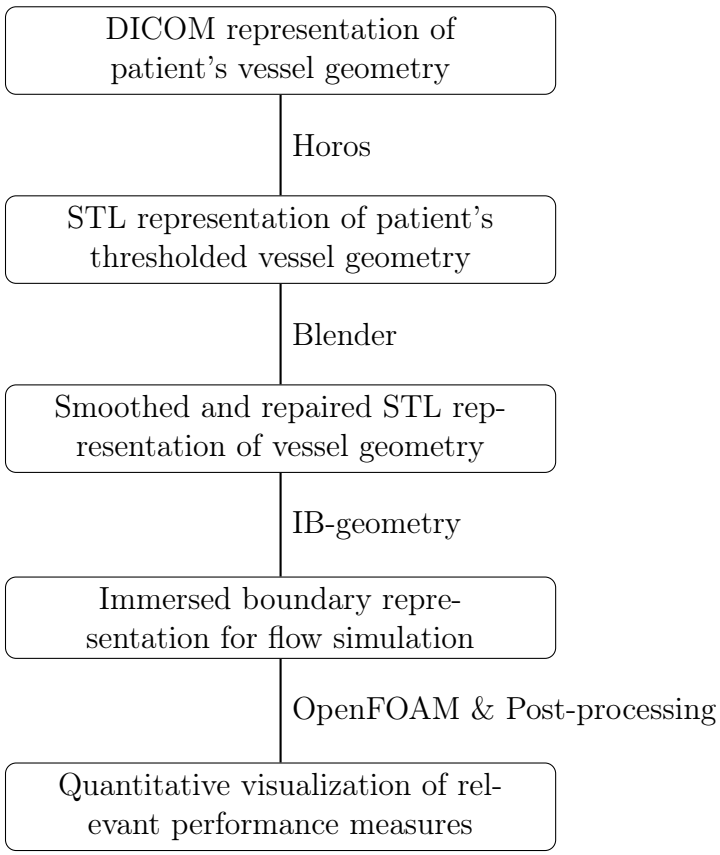


Figure 1.1: Sketch of a workflow for the fluid-mechanical analysis of (stented) aneurysms, connecting patient data in DICOM format to the visualizaton of flow and forces in the region containing the aneurysm.

geometry. As such, the simulations are done not only to investigate the current state of the flow in a diseased vessel, but also to approximate the effect of implanting stents as a feasible treatment option. Additionally, we compare how two different stents, a 'flow-diverter' and a relatively new device called eCLIPs, affect the flow in the aneurysm. The comparisons are presented in detail for a model side-wall aneurysm geometry under both steady-state and pulsatile flow conditions. The possibility of extending the model to a real patient geometry are discussed with illustrations of steady-state and pulsatile flow.

As with any fluid-mechanical analysis, numerical accuracy and efficiency are crucial for the success of the method. The accuracy is determined in part by the adopted spatial and temporal discretization method, and in part by the corresponding spatial and temporal resolutions. In order to accurately resolve all structures in a flow, the smallest characteristic features need to be covered by a certain minimal number of grid cells. Coarsest representations for laminar viscous flow suggest at least 4 grid cells covering one such smallest characteristic feature - higher resolutions than this lower bound would be desired for additional accuracy. We may use this to estimate a rough general resolution requirement that needs to be adhered to:

- In cases of a natural aneurysm that formed in the brain the 'feeding' blood vessel is typically about 3 mm in diameter. Likewise, an aneurysm sac can take sizes in

this range. To resolve laminar flow on these scales accurately, grids with 4 grid cells across a diameter only capture the main features while adopting about 16 grid cells per diameter was found already quite adequate for reliable predictions [15] - this corresponds to a workable coarsest resolution of about 5 grid cells per mm for unstented aneurysms.

- For stented aneurysms a new 'smallest characteristic feature' appears, i.e., the thickness d of the wires/struts that constitute the flow diverter. If we require resolutions of about 4 grid cells per d, this leads to rather high numbers of grid cells and correspondingly much higher simulation times. In fact, for actual flow diverting devices $d \approx 0.1mm$, suggesting that a minimal resolution of 40 grid cells per mm should be adhered to if also the flow around each individual wire/strut should be captured.

  The actual influence of the spatial resolution on the final accuracy can be inferred from systematic grid refinement. We will frequently adopt this strategy in this study. In highly viscous flows as considered here, we observed that well enough away from the stent location, also grids with much lower spatial resolution, e.g., 10 grid cells per mm yield recognizable results, e.g., considering predictions of the velocity profiles. In the close vicinity of the stent deviations of course remain, until proper resolution is applied to also resolve local flow structures here. We will illustrate this in detail for models of side-wall aneurysms.

The organization of this thesis is as follows. In Chapter 2, the medical and clinical motivation for the study is discussed, focusing on the advancement in medical technology that both allows for and demands simulation studies. The description of available geometries, an extensive step-by-step artificial geometry generation in Blender, and a general procedure of handling geometries extracted from medical imaging are discussed in Chapter 3. Chapter 4 briefly describes the Navier-Stokes equations solved in this study, sketches the general idea of the Immersed Boundary Method, and elaborates on the details of setting up and running cases in OpenFOAM. The validation of IBM in OpenFOAM is given in Chapter 5, and the flow prediction and analysis for a model side-wall aneurysm are described in details in Chapter 6. Concluding remarks and an outlook for further research are given in the last chapter.

Chapter 2

Medical and clinical motivation

Aneurysms are a medical condition where an enlargement of a section of artery occurs due to the weakening of the artery wall. In the human brain, aneurysms typically occur in vessels belonging to the Circle of Willis [7, 13, 23], the ring-like arterial structure which supplies blood to the brain. An intracranial aneurysm, occurring within the skull, is especially dangerous since its rupture can cause subarachnoic haemorrhage, the bleeding into the space surrounding the brain, that is often fatal. Half of the patients with aneurysmal subarachnoic haemorrhage who survived, are left with long term neuropsychological effects which decrease the quality of life [13]. As the progress in medical imaging allows practitioners to detect unruptured aneurysms, preventive intervention measures, as opposed to a conservative approach [7], have been developed to address this situation.

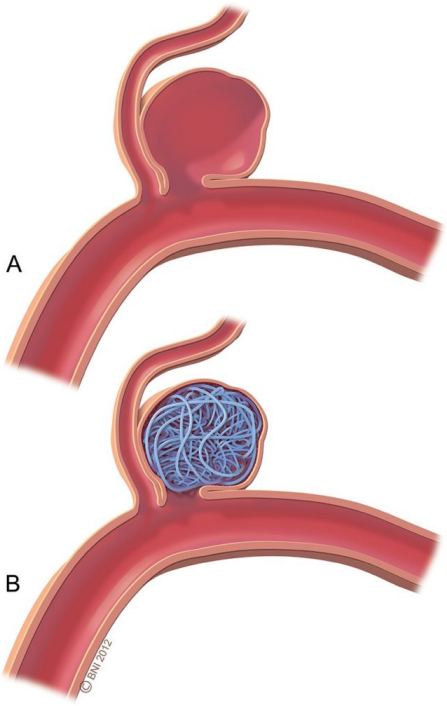


Figure 2.1: Coiling treatment, whereby a metal coil is delivered into the aneurysm sac [5].

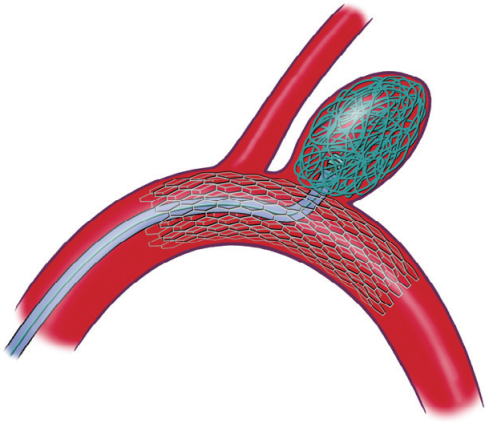


Figure 2.2: Stent-assisted coiling, for aneurysms with wide neck [7].

Interventional treatments for unruptured aneurysm that are regularly used now include surgical clipping and endovascular treatment. Due to its less invasive nature and versatility in addressing a variety of situations, the endovascular treatment has become the more preferable option. Endovascular treatment ranges from stand-alone coiling, stent-assisted coiling, to flow diverter stenting [6, 7]. The type of treatment prescribed for a patient depends on different factors, categorised as the aneurysmal factors - location, size, morphology, etc. - and the patient factors - age, gender, medical and family history, etc. [7]. At its core, the purpose of endovascular treatment is to prevent the pulsatile blood flow from entering thus asserting pressure in the aneurysm sac, known as the water hammer effect, either by filling the sac (coiling) or diverting the flow away from the sac (flow diverter stenting). The expected goal is for thrombosis and occlusion of the aneurysm to occur, whereby the blood flow can be restored as close to normal as possible and the vessel repairs itself through endothelial growth [6, 22].

While the most commonly prescribed treatment is coiling, there has been significant development of flow diverter stenting, with different types of stent manufactured to more effectively address the different types of aneurysm [6, 22]. Flow diverting is generally viewed as the more versatile option, as it addresses a variety of conditions for which coiling is less suitable (i.e., aneurysm with large neck-to-height ratio and aneurysm that occurs adjacent to or at tortuous arteries [6, 7, 22]). Most flow diverters (FDs) are of a hollow cylindrical form with porous wall formed by self-expandable, braided, mesh metal [24], designed in such away to divert the blood without causing too much thrombosis at the artery (as opposed to at the aneurysm sac, which is a desirable outcome).

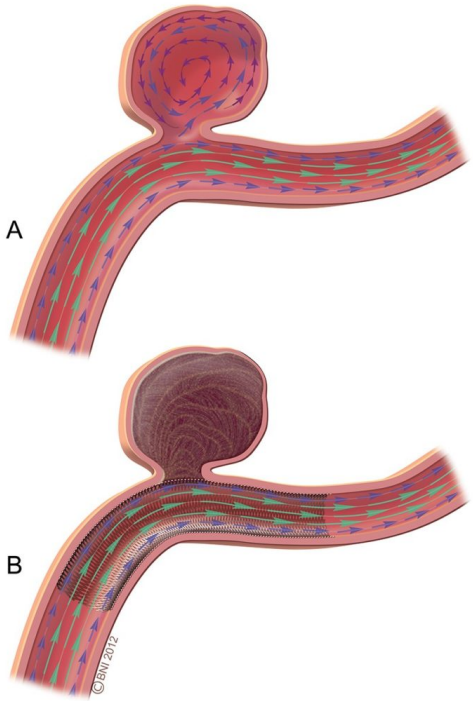


Figure 2.3: Flow diverter stenting, in which a stent is implanted so that the flow entering the aneurysm sac (A) is diverted away fostering normal flow in the vessel (B) [5].

These FDs have limitations, especially regarding the type of aneurysm occurring at the arterial bifurcation ("T-" or "Y-" shaped aneurysm, appearing adjacent to parts of arteries where multiple branching occurs). Bifurcating aneurysm accounts for 64% of aneurysm cases and is harder to treat, as the FDs implanted could cause some blockage into one of the branches of vessel involved [9, 19]. To address this specific issue, a device called eCLIPs (Endovascular Clip System) was designed.

The structure of eCLIPs is non-circumferential, allowing both open access to the flow in all branches and for a better wall apposition. It consists of the flow-disrupting segment and the anchor segment, with a spine rib design [9]. This structure is ideal to be used as a coil assisting device, as well as for producing a flow diverting effect. Additionally, a preclinical study [14] shows that eCLIPs is effective in encouraging endothelial growth by which the vessel repairs itself.

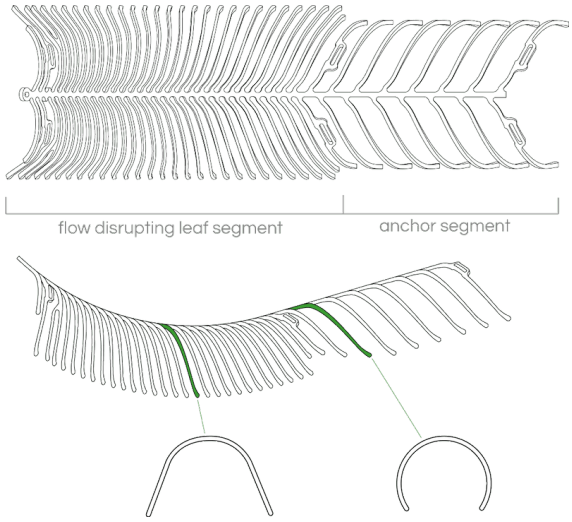


Figure 2.4: The structure of the eCLIPs.

Figure 2.5: The position of the eCLIPs in a bifurcation aneurysm, in which the device allows for free flow in all branches of the vessel [3].

The manufacturing of eCLIPs is relatively new, however, and as such, there have not been many studies investigating the effectiveness of the device. In this study, a numerical tool is used to perform a Computational Fluid Dynamics study to enquire into how the implantation of eCLIPs affects the blood flow in comparison to an earlier-established FD device.

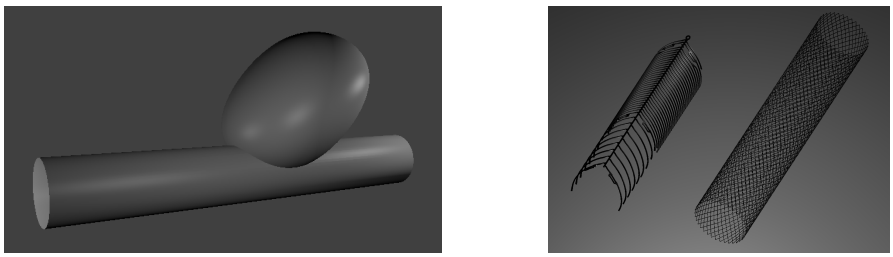
Chapter 3

Geometry handling

The crucial initial step in producing a flow prediction that is patient-specific is properly changing the format of medical data gathered from a patient into a geometry representation upon which the chosen procedure of fluid dynamics analysis can be done. In this study, this process comprises changing the medical data in the form of DICOM (`.dcm`) datasets into properly defined `.stl` files, from which domains of computation are produced for the Immersed Boundary method. Obtaining well-processed medical imagery of real patients' aneurysms is very involved and time-consuming, however, and so this study is first done on geometries that are either artificially created or readily available from previous study as intermediate steps. In this chapter, such geometries will be described (Section 3.1), along with detailed explanations on generating models using Blender as 3D creation suite (Section 3.2) and the general procedure of isolating a target patient-specific geometry from a DICOM dataset (Section 3.3).

3.1 Available geometries

The simulations studying the flow are first done in an artificial geometry intended to mimic a straight vessel with an aneurysm (fig. 3.1(a)) with two implants, a Flow Diverter (FD) and an eCLIPs (fig. 3.1(b)), which were provided by the Department of Neurosurgery, Radboud UMC, Nijmegen.



(a) (b)

Figure 3.1: (a) A straight vessel with aneurysm,
(b) The two implants: eCLIPs (left) and FD (right)

The relatively simple geometry allows for relatively fast and inexpensive simulations which can give a first impression on how the two implants affect the flow inside and around the aneurysm. Once the simple geometry is addressed, the study moves on to focus on a geometry of a diseased vessel extracted from a real patient that was used in a previous study (fig. 3.2) done by Gambaruto et al in 2011 [11].

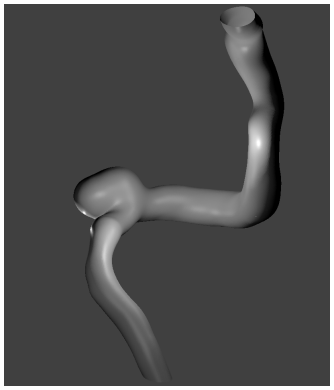


Figure 3.2: A geometry of a diseased vessel extracted from a real patient.

3.2 Geometry generation

For the purpose of creating an artificial geometry and to install the implants in available geometries, the 3D creation software **Blender** was chosen. Blender is a free and open source 3D creation suite which accommodates the entire 3D pipeline, from modelling, rigging, animation, simulation, rendering, compositing, to motion tracking, and even covers video editing and game creation [4]. It runs on multiple platforms- Linux, Windows, and Macintosh, and hence is very versatile. This section describes the steps in the geometry generation using Blender, concentrating specifically on its modelling feature. In particular, the geometry created is a bifurcating aneurysm implanted with both FD and eCLIPs.

3.2.1 Setting up the software

Figure 3.3: The start-up scene of the Blender editor

The installer of the software can be downloaded from the website `www.blender.org`, which will then give prompts to properly install Blender depending on the platform used. The version used for this study is Blender 2.79, and there are two applications installed into the Macintosh platform: the Blender editor itself and the Blender player. For generating the geometry we use the Blender editor.

The default start-up scene is depicted in fig. 3.3, and by clicking anywhere outside of the `splash screen` box, we get the display as depicted in fig. 3.4. The window now shows three main areas: 1) the `3D view port`, which is indicated by the red box, 2) `the outliner` in the green box, and 3) the `properties editor` in the blue box. There are two other parts that are not boxed, the `info editor` at the top and the `timeline` at the bottom. For this study, we do not interact much with these parts.

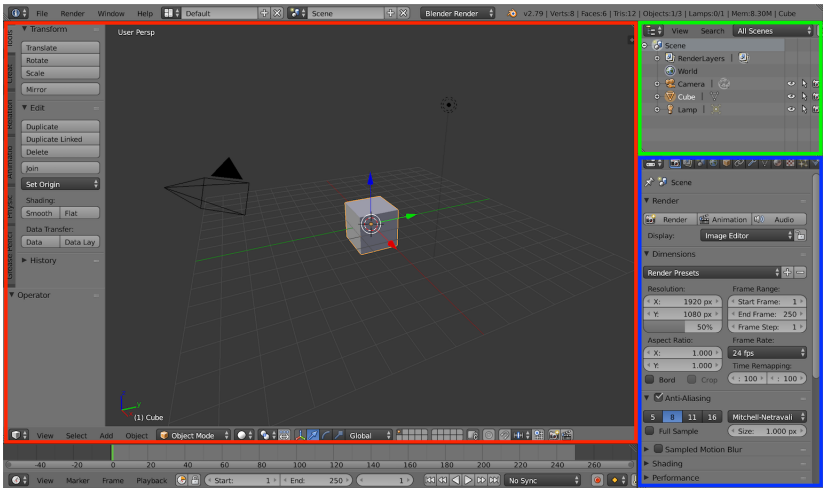


Figure 3.4: The default scene of Blender, showing its main areas

3D view port

The view port is where we create the 3D model, and it has several regions, as shown in fig. 3.5.

1. The red box is the `tool shelf`. For this study the important contents of the tool shelf are the transformation and editing tools and the creation tools, found in the `Tools` and the `Creat` tabs respectively.

2. The purple box is the `operator panel`. When creating an object, the operator panel is where we originally control the dimension, position and subtype of the object.

3. The green box is the `properties region`. This is where we either see the details of the changes made to an object or where we set the changes to the object by manually changing the details. In the default scene fig. 3.4, this region is hidden, and can be displayed by either clicking the plus (+) sign at the top right corner of the main region, or by pressing the keyboard for letter 'N'.

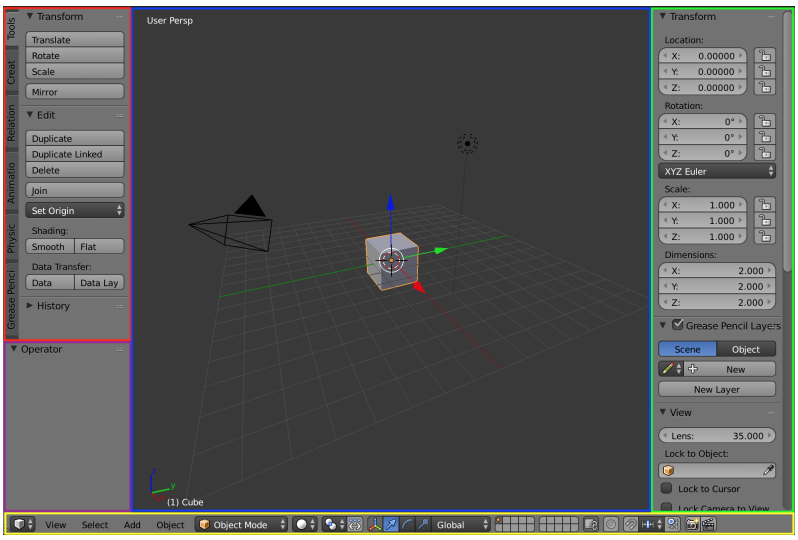


Figure 3.5: The main parts of the 3D view port

4. The yellow box is the `header`. Accessing the `View` tab of the header allows us to change point of view of the main region. In the header, we also find the current mode of the main region, where we switch between `Object Mode` and `Edit Mode`. Another important tab is the `Viewport Shading` where we switch between `Solid` and `Wireframe`, among others. We also access the Transformation properties- orientation and type, in the header.

5. The blue box is the main region. The main region is where we create and manipulate the object.

Outliner

The outliner contains the list of objects that are created and can be seen in the 3D main view port. As a default, Blender creates three objects: the camera, the cube, and the lamp. The camera and lamp are useful for rendering, and are used mainly when there is a need to show the geometry created as a clean image.

Properties editor

The properties editor is used to edit data in the active scene and object. For this study, we mainly use the four `Object` tabs which are the tabs with cube, chain (constrain), spanner (modifier), and triangle symbols.

3.2.2 Navigating the main view port

It is generally advised that when using Blender, or any other 3D creation software, the user should use a three-button mouse to help with navigating the view port. In this geometry generation description, however, we only use the trackpad and shortcuts.

- For rotating the view, we use the *scrolling gesture*.
- For panning the view (up-down, right-left), we use *shift* + *scrolling gesture*.
- For zooming in and out, we use *control* + *scrolling gesture*.
- Activating the number pad is also useful. To do this we go to `info editor`, and click `File` → `User Preferences` → `Input` and check the box for `Emulate Numpad`. By going to the `header` and clicking `View`, we can see the list of shortcut command executed by each number. The numbers that are not listed are for rotating by increments (2, 4, 6, and 8) and switching to the opposite view (9).
- To select an object as the active object, by default we use the *mouse right click gesture*. Using the *mouse left click gesture* to click on the view port will move the `cursor`, the red-and-white circle found at the origin by default.
- Other shortcuts that are useful include:
  - Keypads 'T' and 'N' display and hide the regions on the right and left sides of the main view port respectively
  - Keypad 'X' deletes an object
  - Keypads 'G', 'S', and 'R' translate, scale, and rotate the active object, respectively. To specify the axis for those transformation, we click 'X', 'Y', and 'Z' accordingly. It can also be done manually by dragging the `3D manipulator` - the blue-green-red-legged circle in the middle of the cube in fig. 3.5
  - *shift* + 'D' duplicates the active object
  - *shift* + 'S' activates the choices for snap
  - *command* + 'Z' is the undo shortcut
  - *shift* +*command* + 'Z' is the redo shortcut
  - Keypad 'A' selects all visible objects
  - To quickly switch between `Object Mode` and `Edit Mode` we use `tab`

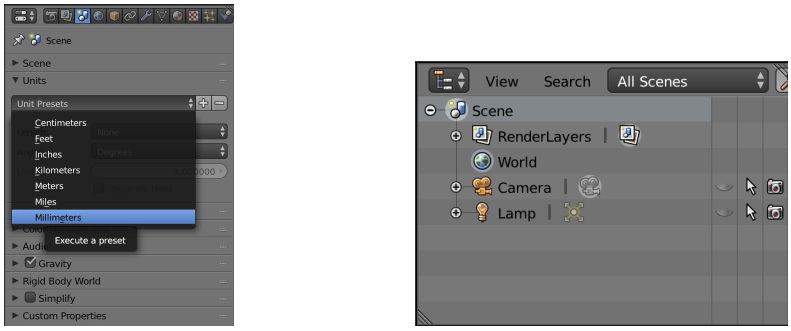
3.2.3 Creating a model object: bifurcating aneurysm

The followings are the steps to create the bifurcating aneurysm.

Preparing the scene

1. We first set the scene to have the dimension desired, which is in millimetre. On the `Properties region`, we choose the `Scene` tab, and under `Units` we choose `Millimeters` (fig. 3.6(a)).

2. We do not need any of the objects created by default, and so we delete the Cube (keypad 'X') and hide the Camera and Lamp by clicking the eye symbol next to them in the `Outliner` (fig. 3.6(b)).



(a) (b)

Figure 3.6: (a) Setting the scene unit, and (b) hiding objects in Blender.

Creating the main vessel

1. To mimic the vessel, we create a cylinder. The first cylinder we make is the *main vessel*, with diameter 4mm and length 8mm. We go to the `Tool shelf`, click the `Creat` tab and select `Cylinder`; every new object appears at the position of the `cursor`, which by default is at the origin.

2. In the operator panel we set the `Radius` to 2mm, the `Depth` to 8mm, and the `Cap Fill Type` to `Nothing`.

3. For better tracking, rename the Cylinder in the `Outliner` as MainVessel. The resulting window is seen in fig. 3.7.

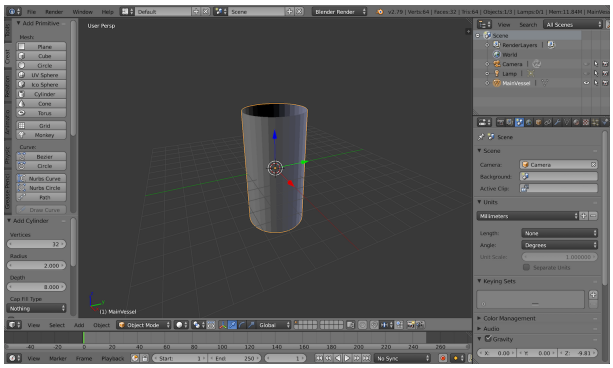


Figure 3.7: Creating cylinder as the main vessel.

Creating the branch vessel

1. The *branch vessel* is also a cylinder, but one with smaller diameter at 3mm and greater length, at 16mm. The steps are similar to the creation of the main vessel. Rename this cylinder BranchVessel.

2. We rotate the branch vessel by 90° in the $x-$direction, using shortcuts keypad 'R' $\rightarrow$ keypad 'X' $\rightarrow$ keypads '9' and '0'.

3. We then translate the branch 5.5mm upwards (positive $z-$direction) so that it sits at the top of the main vessel, using shortcuts keypads 'G' $\rightarrow$ 'Z' $\rightarrow$ '5', '·', and '5'. The right ortho [1] view of the view port can be seen in fig. 3.8.

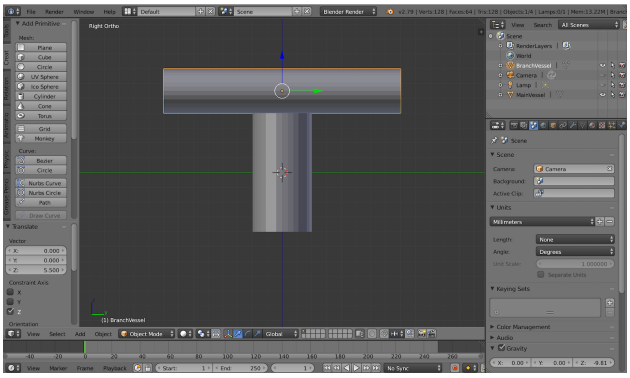


Figure 3.8: Creating and setting the branch vessel.

Creating the aneurysm sac

1. We then create a sphere with diameter 5mm to mimic the *aneurysm*. We choose the `Ico Sphere` from the `tool shelf` and set the `Subdivision` at 4 and the `Size` at 2.5mm. Rename this sphere as Aneurysm.

2. We translate the aneurysm 8.5mm upwards so that it sits at the top of the branch with some overlapping part. The right ortho view of view port can be seen in fig. 3.9.

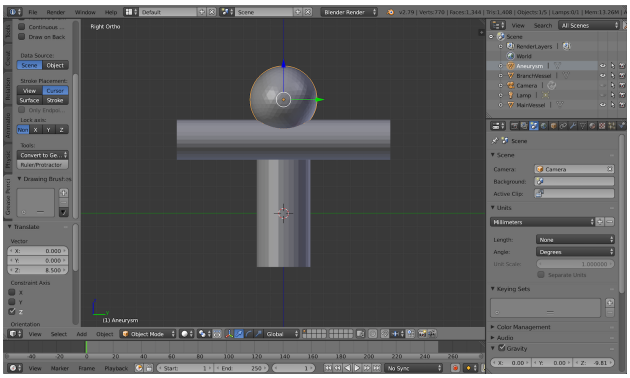


Figure 3.9: Creating and setting the aneurysm sac.

[1] the `Ortho` view gives a flat, 2D view of the scene, as opposed to the real perspective given in `Persp` view. We switch to and out of Ortho view by pressing keypad '5'.

Modifying the objects

1. We now modify the objects, for which we have to go to `Edit Mode` by pressing *tab*.

2. We first extend the main vessel so that the extension covers the part of the branch where the two objects should merge. In `Edit Mode`, we select the main vessel. By default, the part of the object selected are the `Vertex`, and we need to change that into the `Edge` by pressing *control* + *tab* and selecting `Edge` from the options.

3. Click on one of the edges at the top of the main vessel, and press *option(alt)* and click another edge simultaneously to select all the edges in the loop.

4. While all the edges are selected we press the keypad 'E' for `Extrude` followed by keypads 'Z' and '0', '·', '7', and '5' for extruding 0.75mm upwards. We then scale it by the factor of 0.9, using shortcuts keypads 'S' → keypads '0', '·', and '9'.

5. We execute two more `Extrude-Scale` alterations, the first 0.5mm upwards scaled to 0.9 and the last 0.25mm awards scale by the factor of 0.9. The right perspective now looks like fig. 3.10.

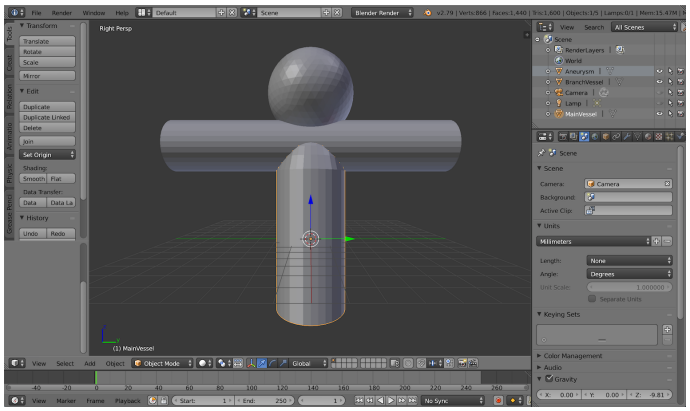


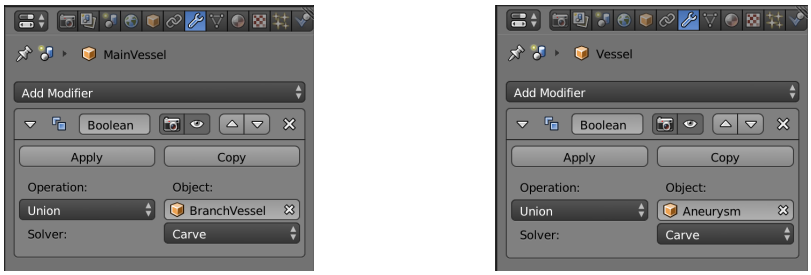
Figure 3.10: The result of extruding the main vessel to connect it with the branch vessel.

Merging the objects

1. We can now merge the objects. We first merge the main and branch vessels. We switch to `Object Mode` and select the main vessel.

2. On the `Properties region`, we choose the `Object Modifier` tab and under `Add Modifier` we select `Boolean`. We set the details as seen in fig. 3.11(a) and click `Apply`. We can now hide the branch vessel and have the now joined main and branch vessel renamed as Vessel.

3. For merging the vessel and aneurysm we do the similar alterations: select the vessel, add Boolean modifier with details as seen in fig. 3.11(b), and then hide the aneurysm.



(a) (b)

Figure 3.11: The `Boolean` modifier setups for combining (a) the branch vessel to the main vessel, and (b) the aneurysm to the rest of the object.

We now have the bifurcation aneurysm and we rename Vessel as Bifurcation. The resulting window is seen in fig. 3.12.

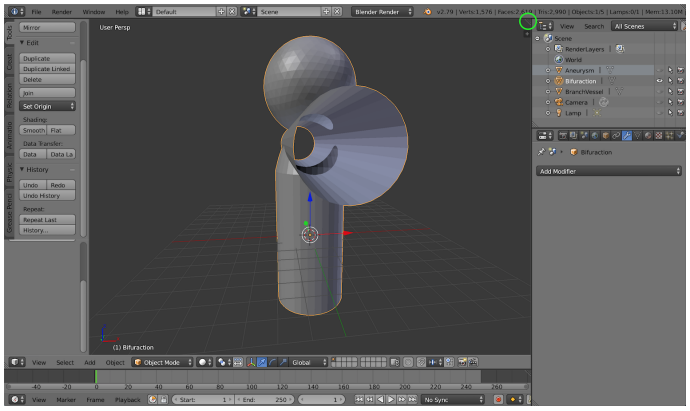


Figure 3.12: The whole bifurcating aneurysm after the application of the Boolean modifier.

We safe this blender file by pressing *control* + 'S', and naming the file `bifurcation.blend`.

Duplicating the object

1. As we will need three different bifurcation geometry, each for the geometry without implants, with FD, and with eCLIPs, we duplicate the object Bifurcation.

2. We do this by selecting bifucation, pressing *shift* + 'D', and translating the duplicates 10mm away in the $x-$direction both ways. Rename the two new bifurcations as BifurFD (the one positioned at +10mm) and BifurEclips (the one positioned at -10mm). We hide both of these new objects for now.

Accommodating for IBM

1. Concentrating first on the original bifurcation object, we accommodate for the Immersed Boundary Method (explained in Section 4.2), by encasing it with a box.

We do this by creating a `Cube` from the `tool shelf`, setting the `Radius` to 2.5mm, which is the largest radius of the bifurcation, belonging to the sphere of aneurysm.

2. We scale up, down and sidewise to make sure the box cover the entire bifurcation. The easiest way to do this is by displaying the `Properties region` (press the keypad 'N'), setting the `Dimensions` to X: 5mm, Y: 16mm, and Z:15mm and then translating it 3.5mm upwards. Rename this cube BoxBifur.

3. We now apply the Boolean modifier to the BoxBifur, setting `Operation` to `Difference`, `Object` to `Bifurcation`, and `Solver` to `BMesh`. Hide the bifurcation. A split view [2] of the solid and wireframe of BifurBox can be seen in fig. 3.13. We can now hide this BoxBifur.

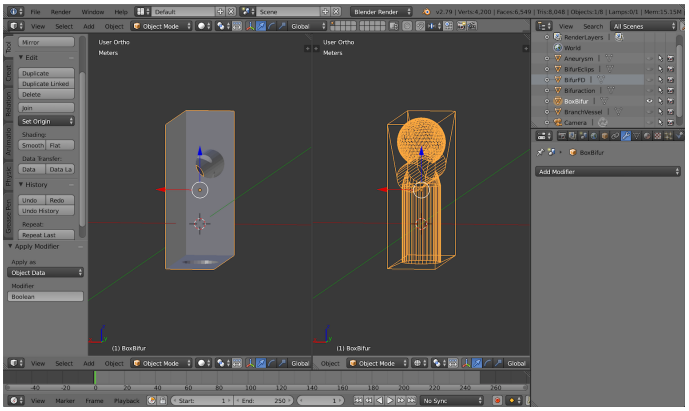


Figure 3.13: The split view of the bifurcating aneurysm placed in a box, showing the solid (left) and wireframe (right) views.

Importing the implants file

1. For this study, the two implants, the *FD* and *eCLIPs*, have been provided as an `.fbx` file, and can be found here. Download the file.

2. To import the file into the active scene, we go to the `info editor`, and click `File` → `Import` → `FBX(.fbx)`, locate the file and click `Import FBX`.

3. We now have two more objects added, the FlowDiverter and eCLIPs. To start with, we hide the FlowDiverter.

Modifying the eCLIPs

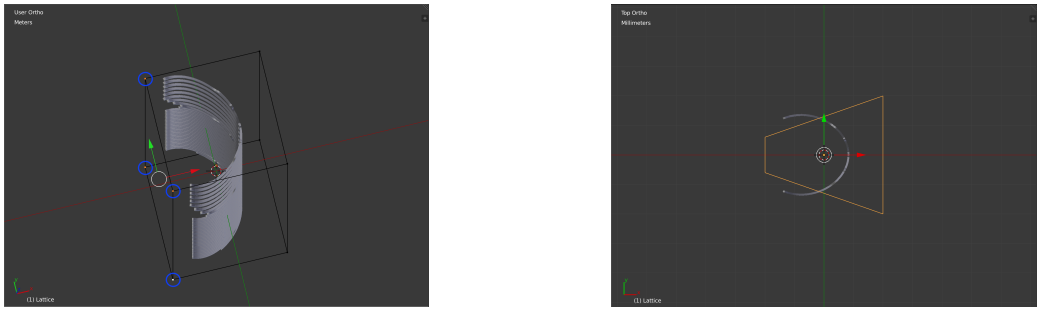
1. The feature of eCLIPs is that one size should be flexible enough to fit the any size of vessel. The original eCLIPs from the imported file is set to fit a 4mm in diameter vessel. Since the branch vessel where we are inserting the eCLIPs is 3mm in diameter, we need to modify the eCLIPs.

[2] To have split view of any areas we click and drag the right corner of the area, as indicated by the green circle in fig. 3.12.

2. We do this by using the `Lattice Deform modifier`. With eCLIPs located at the origin and selected, we go to the `tool shelf` and create `Lattice`, setting the `Radius` to 4mm. We scale the lattice by the factor of 4 in the $z-$direction, and then translate it 1mm upwards. The lattice should now encases the eCLIPs.

3. With the eCLIPs selected, on the `Properties region`, we choose the `Object Modifier` tab and under `Add Modifier` we select `Lattice`. Set the `Object` as `Lattice`.

4. With the lattice selected, we switch to `Edit Mode`, and select the four vertices at the opening side of eCLIPs, as seen in fig. 3.14(a). We scale these vertices in the y-direction by the factor of 0.3. The top ortho view now looks like fig. 3.14(b). We can now apply the modification by selecting eCLIPs, and clicking `Apply` under the `Object Modifier` tab.

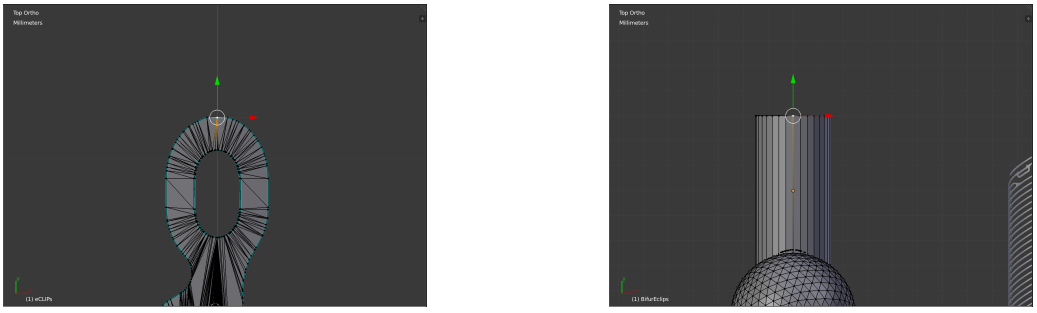


(a) (b)

Figure 3.14: (a) Selecting the four vertices of the lattice, (b) Top ortho view of the modified eCLIPs.

Snapping the eCLIPs

1. Rotate the eCLIPs 90° in the $x-$direction and -90° the $y-$direction to put it in the correct orientation.



(a) (b)

Figure 3.15: Selecting the vertices (a) on the tip of eCLIPs, and (b) on the bifurcating aneurysm object.

2. Switch to `Edit Mode`, and choose `Vertex`. Select the vertex that is seen in fig. 3.15(a), and make it the place for the `manipulator` for the eCLIPs. We do this by pressing *shift* + 'S' selecting `Cursor to Selected`. Switching back to `Object Mode`, we press *shift* + *control* + *option(alt)* + 'C' and select `Origin to 3D Cursor`.

3. Select the BifurEclips object and switch to `Edit Mode`. Select the vertex as seen in fig. 3.15(b), and make it the place for the `manipulator` for the BifurEclips. The procedure is the same as above.

4. Leave the 3D cursor at the vertex in BifurEclips. Select the eCLIPs and press *shift* + 'S' selecting `Selection to Cursor` to snap the eCLIPs into BifurEclips. We then translate the eCLIPs 2.5mm in the negative $y-$direction, so that the flow diverting part of the eCLIPs cover the entire opening of the aneurysm.

5. We have to modify the BifurEclips a little, since the anchoring part of the eCLIPs goes outside of the branch vessel. Select BifurEclips and switch to `Edit Mode`. Select the edges in a loop of the ending of branch vessel where the eCLIPs sticks out. Translate the edges by 3mm in the negative $y-$direction. Switch back to `Object Mode`.

6. We now perform the Boolean Modifier, selecting first the BifurEclips, and setting `Operation` to `Difference`, `Object` to `eCLIPs`, and `Solver` to `BMesh`. We can now hide the eCLIPs. The split view of the BifurEclips now look like seen in fig. 3.16.

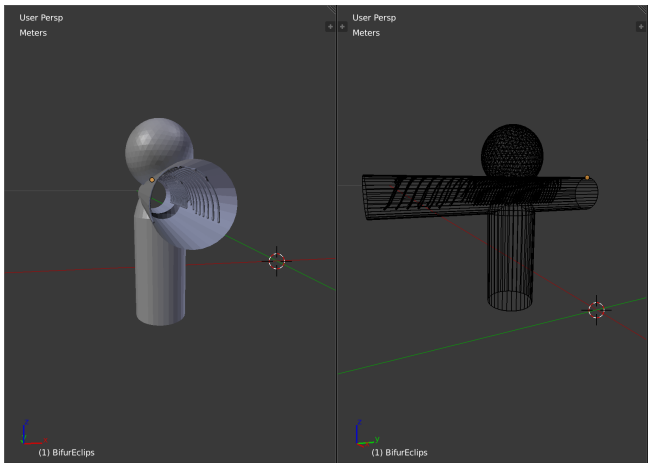


Figure 3.16: The split view of the bifurcation aneurysm implanted with eCLIPs.

7. We do the same for BifurEclips as we do in Section 3.2.3. For neatness, we first press *shift* + *control* + *option(alt)* + 'C' and select `Origin to Geometry`

   (a) We create a `Cube` with `Radius` 2.5mm, snap it to BifurEclips, and rename it as BoxBifurEc. On the `Properties region`, set `Dimensions` to X: 5mm, Y: 19mm, and Z:15mm. Translate it by 2.8mm in downwards and by 0.62mm in the negative $y-$direction.

(b) We now apply the Boolean modifier to the BoxBifurEc, setting `Operation` to `Difference`, `Object` to `BifurEclips`, and `Solver` to `BMesh`. We can now hide the object bifurEclips. A split view of the nee BoxBifurEc can be seen in fig. 3.17. We can now hide this BoxBifurEc.

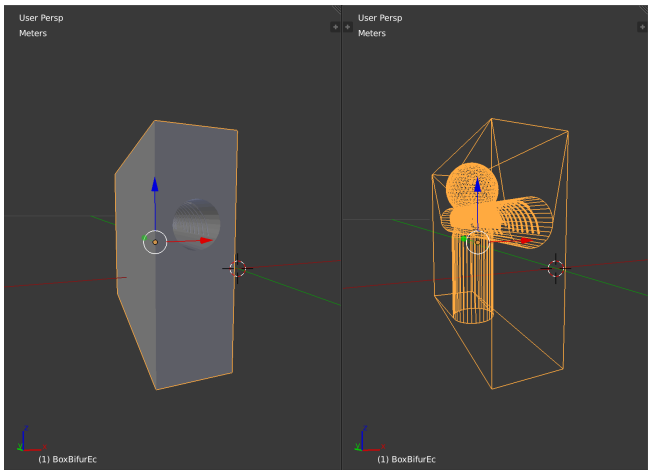


Figure 3.17: The bifurcating aneurysm implanted with eCLIPs placed in a box.

Modifying and snapping the FD

1. Just like with eCLIPs, the FD from the `.fbx` file is designed to fit vessels with diameter 4mm. The dimension of this FD is then 4mm in diameter with length 20mm. We will scale the FD in all directions by a factor of 0.75. Unhide the FlowDiverter and on the `Properties region` set the `Dimensions` to X: 3mm, Y: 3mm, and Z: 15mm or scaling it by 0.75.

2. Rotate the FD by 90° in the $x-$direction to put it parallel to the branch vessel.

3. We will now snap the FD first. Unhide BifurFD, then we follow the steps 2, 3, and 4 from Section 3.2.3 to define the origins to snap. The vertices selected from the FD and BifurFD are shown in fig. 3.18; pay attention to the view (`Top Ortho`) and make sure to select the vertices that have corresponding positions.

4. Move the origins back to the centre of mass in each object by pressing *shift* + *control* + *option(alt)* + 'C' and select `Origin to Geometry`. Select the FD and translate it 0.5mm in the positive $y-$direction.

5. We will now bend the FD to fit the shape of the bifurcation.

   (a) Set the view to `Right Ortho`, so that it is easier to control the bend. Move the 3D cursor to FD (*shift* + 'S' then `Cursor to selected`).

   (b) For bending, we enter the `Edit mode`, select the entirety of the FD, and put the mouse cursor in the green arrow indicating the $y-$axis. Press *shift* + 'W', and set the `Bend Angle` to 90° and the `Radius` to 1.5.

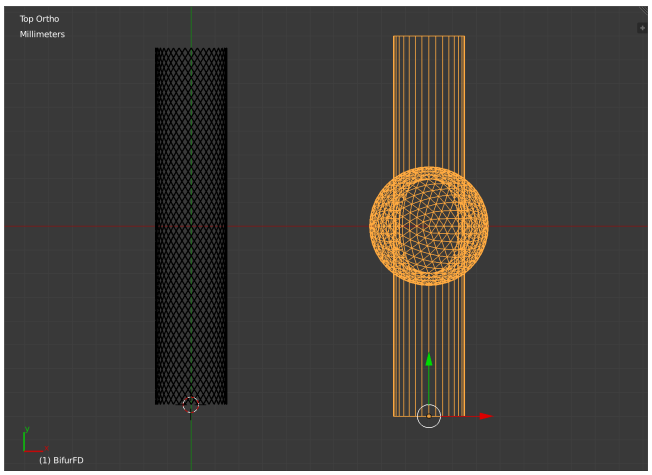


Figure 3.18: Selecting the vertices on the FD (left) and the bifurcating aneurysm object (right)

(c) Translate the FD by 1mm in the negative $y-$direction.

6. We now have the similar problem as in step 5 of Section 3.2.3, and so we can follow steps 5-7 of the section (with translations in step 7 changed to 0.1mm in negative $x-$direction, 1.1mm in the positive y-direction, and 0.6mm downwards), and name the cube BoxBifurFD. The resulting geometry can be seen in fig. 3.19.

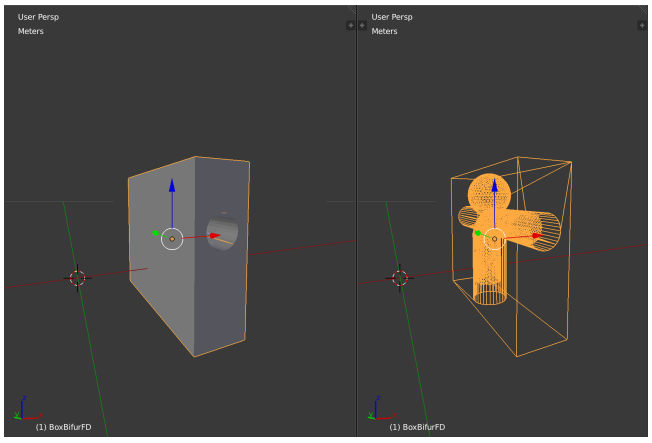


Figure 3.19: The bifurcating aneurysm implanted with FD placed in a box.

3.2.4 Exporting into readable format

To make sure that the objects created in Blender can then be read by OpenFOAM, we need to export it as `.STL` files. Since we have multiple objects, we want each of them to be exported as one object per .stl file. To do this, we execute the following steps:

1. Select the object we want to export, for example, BoxBifurEc.

2. On the `info editor`, and click `File` → `Export` → `Stl(.stl)`.

3. Set the bottom left part of the resulting window as shown in fig. 3.20, and name this file `boxBifurEc.stl`. Make sure the location of the file in the folder is easy to locate.

Figure 3.20: The setup of exporting the object as an `.stl` file.

4. If we want to export all of the objects created each as separated files, uncheck the `Selection Only` option and set `Batch Mo...` to `Object`. The resulting files will be named according to the names given in the `Outliner`.

3.3 The ideal path of geometry handling

When an MRI/MRA image dataset of an aneurysm is available, most probably in `.dcm` (DICOM) format, the ideal path of the geometry handling starts with converting the images into an `.stl` file. A free and open source software to do that is **Horos** [1].

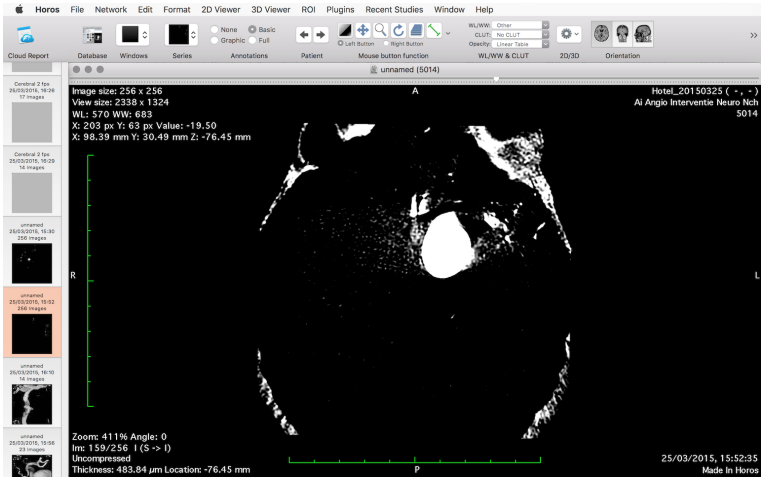


Figure 3.21: A DICOM dataset of a CT-scan of a human brain viewed in Horos.

Figure 3.21 shows a series of `.dcm` data loaded into Horos. This dataset of an anonymous patient was provided by the Department of Neurosurgery, Radboud UMC, Nijmegen. Using the `3D Volume Rendering` tool, the geometry of the aneurysm can be seen as depicted in fig. 3.22

Figure 3.22: 3D Volume Rendering of a real aneurysm

To convert the dataset into an `.stl` file, the tool used is the `3D Surface Rendering`. However, unlike the smooth rendering produced by the `3D Volume Rendering`, the result of Surface Rendering of the original dataset would contain a lot of noise as seen in fig. 3.23. The noise in the structure comes from other tissues and bones that were also recorded during the scan. The region of interest in this case is only the circled part and its immediate surroundings, which is the aneurysm sack, and the part of the vessel before and after its opening.

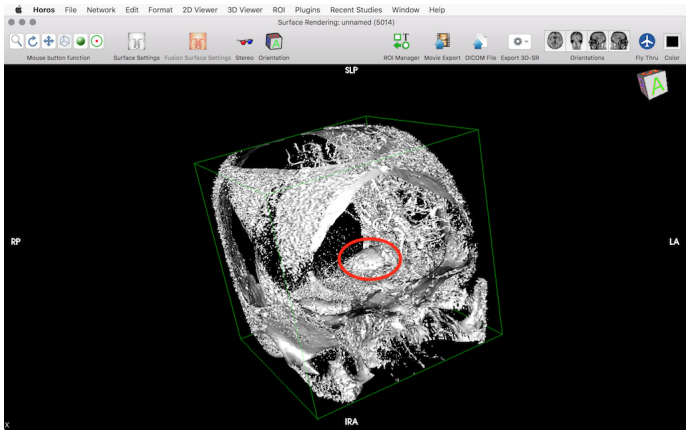


Figure 3.23: 3D Surface Rendering of the multi-valued DICOM data series with noise.

Hence, a segmentation threshold needs to be defined, so that the multi-valued DICOM data series can be first converted into a two-valued series, removing most of the unnecessary noise.

3.3.1 Converting DICOM data series into `.stl` files

1. Firstly, the approximate threshold value is investigated by selecting some areas in the region of interest to display their values. This is done by going to `Mouse button function` in the toolbar, selecting the `Oval` button and by clicking on some of the areas in the region of interest, by which the data values will be displayed automatically. As can be seen in fig. 3.24, the approximate threshold should be in the range of 2000 - 5000.

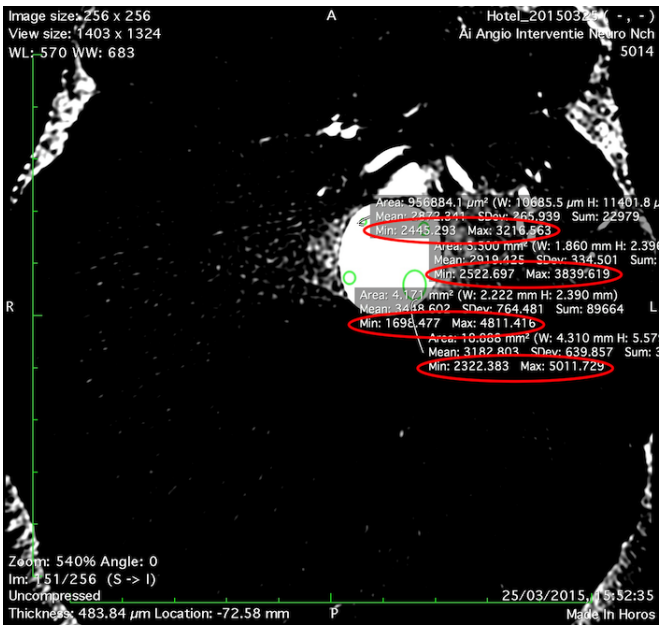


Figure 3.24: The view of the data series, showing the possible values of threshold.

2. In the Menu bar, select `ROI` → `Grow Region (2D/3D Segmentation)...`, and set the `Segmentation Parameters` as seen in fig. 3.25. When clicking `Compute`, a new data series with only two values (0 or 1000) will appear on the right hand side of the original data series. Click only on this image so that it is highlighted red, and then from the `Menu`, choose `3D Viewer` → `3D Surface Rendering`.

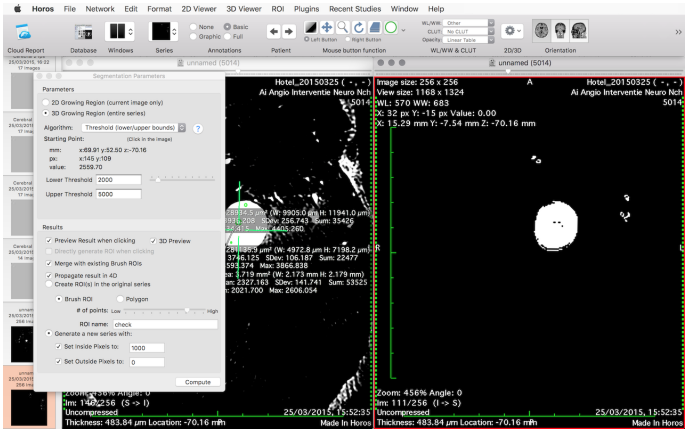


Figure 3.25: The setup of for the `Segmentation Parameters`, and the split view of multi-valued (left) and two-valued (right) series.

3. A prompt to choose a resolution will appear, and after clicking OK, a surface rendering of the segment of interest will be displayed as seen in fig. 3.26.

4. From the `Toolbar` choose `Export 3D-SR` → `Export as STL (.stl)` and name the file accordingly. An `.stl` file ready to be treated with a 3D creation suite, e.g. **Blender**, is now available.

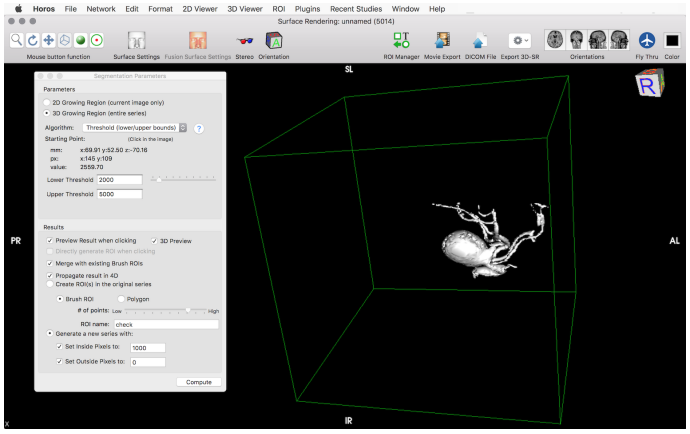
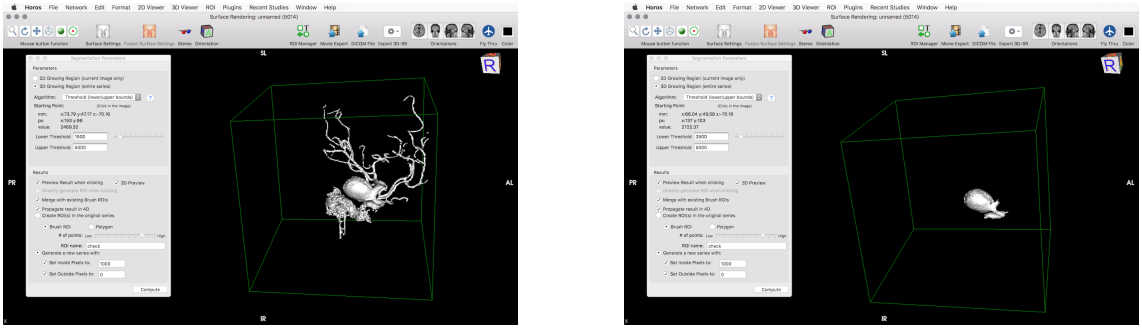


Figure 3.26: The surface rendering of the thresholded, two-valued series.

3.3.2 On choosing the threshold values and resolutions

Removing the noise from data often involve the risk of also removing parts of the region of interest which have the same data values as the noise. Ideally, the surface rendering of the region of interest should be as close as possible to the images captured by the MRI. However, the cost and efficiency of treating the resulting geometry also need to be taken into account.

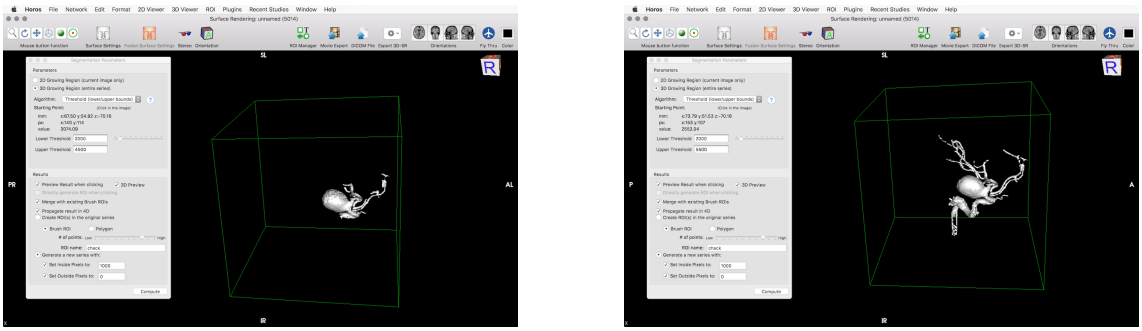
In this case, the threshold values below and above the selected values of 2000 - 5000 would give geometries that are more difficult to treat. Figure 3.27 shows the two geometries resulting from choosing lower threshold that are less (i.e., 1500 fig. 3.27(a)) and greater (i.e., 2500 - fig. 3.27(b)) than the selected value of 2000. The lesser lower threshold gives a geometry that still contains too many noises, while the greater lower threshold gives a geometry that is no longer representative of the the region of interest.



(a) (b)

Figure 3.27: Surface rendering of lower threshold values that are (a) less than, and (b) greater than the chosen value of 2000.

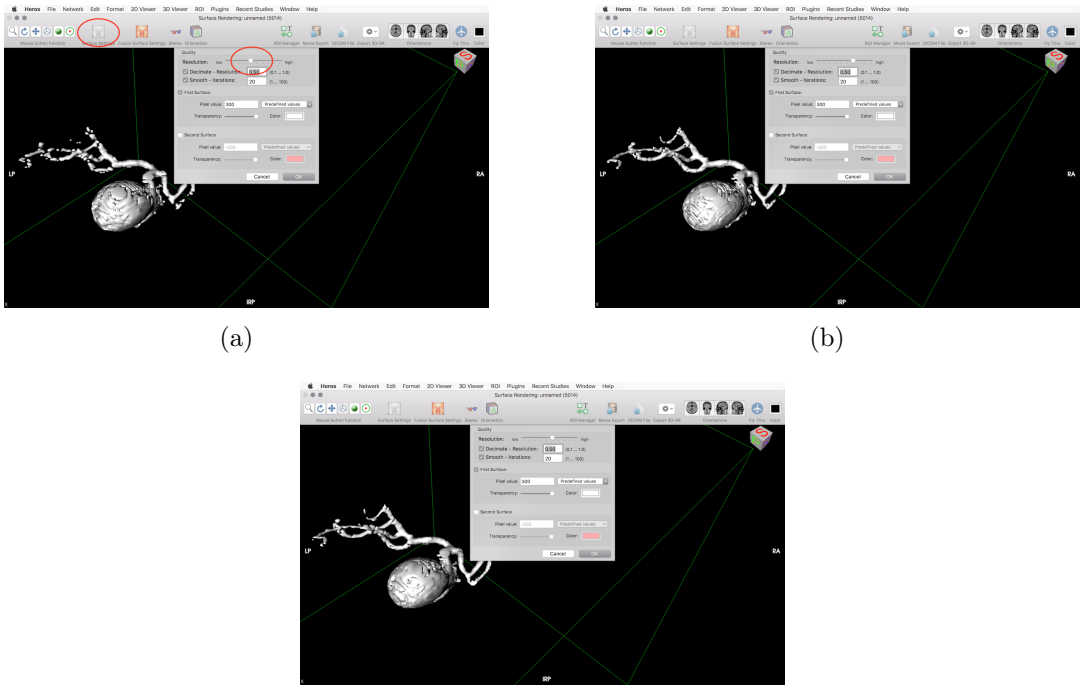
A similar but reversed occurrence is observed when adjusting the upper threshold value. As seen in fig. 3.28, choosing an upper threshold less than the selected 5000 (i.e., 4500 - fig. 3.28(a)) gives a geometry that lost too much information, while choosing a greater value (i.e., 5500 - fig. 3.28(b)) gives a very complex geometry.



(a) (b)

Figure 3.28: Surface rendering of upper threshold values that are (a) less than, and (b) greater than the chosen value of 5000.

Additionally, the choice of resolutions of the surface rendering also impacts, albeit to a lesser degree, the resulting `.stl` geometry. Figure 3.29 illustrates the comparison between the resolution chosen and the resulting geometry. Figure 3.29(a) represents the default resolution set by Horos, which is increased one step the slide bar each time in the other two pictures. In this case, the selected resolution is the one represented by fig. 3.29(c); the default resolution gives vessels with holes, while a higher resolution (fig. 3.29(b)) results in aneurysm sac that is too 'bumpy'. Of course, the final decision should be based on the quality of the flow prediction - ultimately the definition of the geometry should go hand-in-hand with the flow prediction quality.



(a) (b)

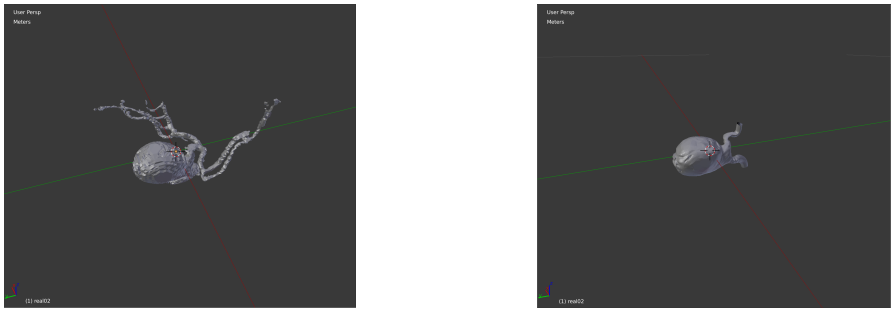
(c)

Figure 3.29: Surface rendering showing the effect of choosing resolution values. (a) The default resolution, (b) higher resolution, and (c) the selected resolution.

3.3.3 Treating the `.stl` file in BLENDER

Admittedly, treating a thresholded `.stl` file of a real geometry is the most difficult and most user-dependent part of the geometry handling in this study. The following is the brief description on how the geometry extracted in fig. 3.26 is prepared for the mesh generation.

1. The `.stl` file is imported into Blender using the same procedure described in Section 3.2. Adjust the position and the center of the geometry, so that what is seen in Blender is as depicted in fig. 3.30(a).



(a) (b)

Figure 3.30: (a) The imported `.stl` file of fig. 3.26 in Blender.
(b) The simplified geometry of fig. 3.30(a).

2. The parts of the geometry that are of interest are the aneurysm sac and a small bit of vessel before and after the sac. Hence, the next step is to erase the parts that are not needed.

3. A combination of smoothing features in Blender and the `extrude` command resulted in a simplified geometry as seen in fig. 3.30(b)

4. Accommodating for IBM, the geometry is placed in a box (fig. 3.31) and then exported as an `.stl` file ready for mesh generation.

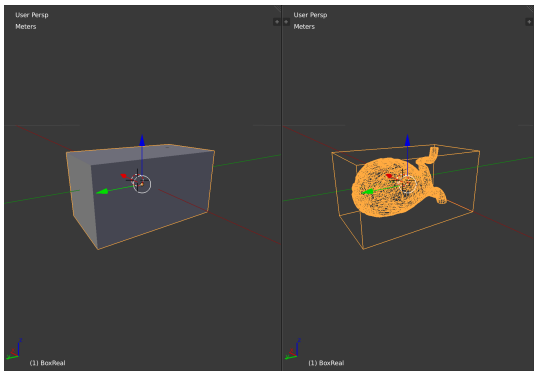


Figure 3.31: The split view of the geometry placed in a box.

Chapter 4

Numerical method

The use of Computational Fluid Dynamics (CFD) in exploring options for treatment of cerebral aneurysms has been foreseen at least since the start of the millennium (Metcalfe, 2003, as cited in [15]). As the advancement of medical imaging and the tools by which the images are processed allow for representative reconstructions of the cerebral vessels and aneurysm geometry of specific patients, numerical modelling and simulations for such geometry have become more constructive and informative. Through CFD, qualitative and quantitative characteristics of blood flow in aneurysms can be presented and analysed, providing a contribution into the intricate medical decision making by the practitioners. In this study, the generated numerical tool uses an Immersed Boundary Method (Section 4.2) solving the Incompressible Navier-Stokes equations (Section 4.1) running on the OpenFOAM platform (Section 4.3).

4.1 Navier-Stokes equations

Among the different approaches for modelling blood flow in the human brain is representing blood as incompressible Newtonian fluid with a constant viscosity (Cebral et al, 2005, as cited in [15]). In the Circle of Willis, a certain Reynolds number (Re) is taken to represent the flow regime. The simulation of the flow is then done by solving the Incompressible Navier-Stokes equations, equipped with the Dirichlet boundary condition, i.e. no-slip condition, at the vessel wall:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \frac{1}{Re} \nabla^2 \mathbf{u} + \mathbf{f} \tag{4.1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{4.2}$$

where $\mathbf{u}$ is the velocity of the fluid, P is the pressure, and Re is the Reynolds number. The forcing term $\mathbf{f}$ is used here to represent the flow domain and is dependent on a masking function discussed in the next section.

4.2 Immersed Boundary Method

As an alternative to the body-fitted approach of representing the geometry an aneurysm, this study adopts the Immersed Boundary Method (IBM) approach. The volume-penalizing IBM allows for direct transformation of complex shapes to be 'immersed' in a Cartesian computational grid, simplifying the process of representing the geometry of the vessel and aneurysm into a discretized computational domain. Additionally, IBM is also capable of reducing the computational cost by reducing the per-grid-point operation count owing to the lack of additional terms caused by grid transformation in curvilinear body-fitted approach [18], and because effective solvers are available on Cartesian grids.

The main feature of IBM is the so-called *masking function*, a binary function which distinguishes the "solid" or the tissue part (assigning the value 0) from the "liquid" part (assigning the value 1)[15]. The resulting representative geometry in our study is that of a form of staircase approximation, which become refined as the grids become finer. With respect to the eq. (4.1), the forcing term $\mathbf{f}$ is given by

$$\mathbf{f} = -\frac{1}{\varepsilon} H(\mathbf{x})\mathbf{u}(\mathbf{x}, t) \tag{4.3}$$

where $H(\mathbf{x})$ is the masking function [15, 16]. If $\mathbf{x}$ belongs to the fluid part of the domain, the masking function $H(\mathbf{x}) = 0$; otherwise, it is 1. A 2D illustration is given in fig. 4.1 [1]. In eq. (4.3) the time-scale $\varepsilon \ll 1$, implying a strong penalty for non-zero velocities in regions where $H = 1$. Stated differently, in the tissue ($H = 1$) a non-zero flow velocity can not be maintained and flow is only allowed in the fluid part of the domain ($H = 0$). We use very low values of ε on the order of 10^{-10} in dimensionless formulation, as adviced in [15].

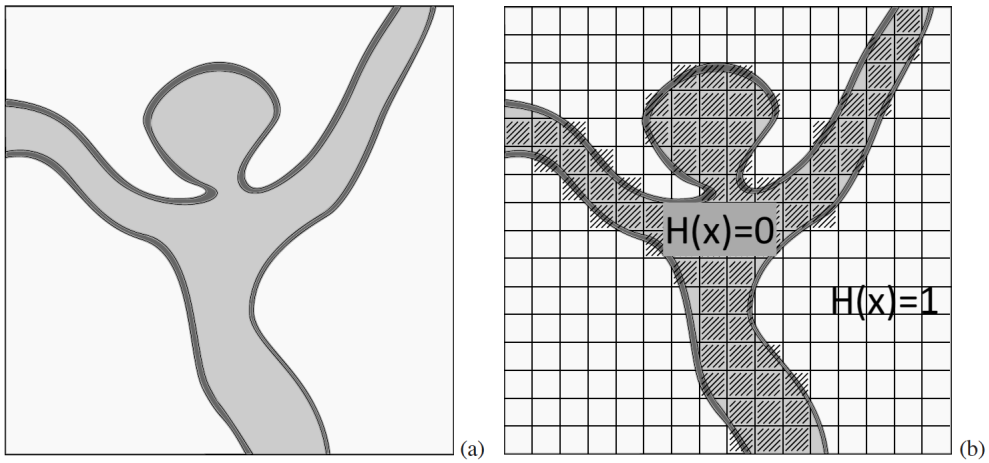


Figure 4.1: 2D representation of IBM, where a complex geometry (a) is 'immersed' into a Cartesian computational grid, with given values for the masking function $H(\mathbf{x})$.

[1] The illustration is modified from [15], p.18.

4.3 OpenFOAM

Open Source Field Operation and Manipulation (OpenFOAM) is an open source C++ toolbox for developing numerical solvers and pre-/post- processing utilities used in solving continuum mechanics problems, including CFD [2]. OpenFOAM was produced as an effort to provide a reliable free and open sourced coding tool for researchers to use that specifically addresses CFD problems [8]. The ease of use, the options of costumization of the solvers, and the existence of a community of users ready to provide free resources are among the advantages of using OpenFOAM.

In essence, running a case in OpenFOAM consists of three main things: preprocessing, solving, and postprocessing [8]. *Prepocessing* usually involves the mesh generation, in which the computational domain is built and checked for completion. The standard utility used for mesh generation is `blockMesh`. In this study, however, due to the implementation of the IBM for domain generation, a specific process implemented in MATLAB replaces `blockMesh`, and the domain is converted into the OpenFOAM format afterwards. We return to the full sequence of processing steps involved in analyzing a particular problem later. *Solving* a case is the part of the process, where the solver, governing equation, boundary conditions, discretization schemes, properties, and control parameter are chosen or defined. In this study, in particular, all the generated solvers are adapted as further developments from the `icoFoam` solver, which is the OpenFOAM solver for incompressible, laminar fluid. *Postprocessing* normally consists of producing graphical outputs, for which OpenFOAM uses an open-source, multiplatform data analysis and visualization application called PARAVIEW. In this study, postprocessing the data also comprises the sampling of the solution that is then plotted in MATLAB.

The following section will describe in more detail the set up of cases and the algorithm used in the OpenFOAM simulations for this study. Unless stated otherwise, the descriptions are adapted from [2].

4.3.1 General set up of OpenFOAM cases

The three directories essential for any OpenFOAM cases are the `system`, `constant`, and `0` (at the start of time directories) directories.

- The `system` directory contains at least 3 files:
  - `controlDict` where run control parameters are set including start/end time, time step and parameters for data output,
  - `fvSchemes` where discretisation schemes used in the solution may be selected at run-time, and
  - `fvSolution` where the equation solvers, tolerances and other algorithm controls are set for the run.

Additionally, for the purpose of running the simulation in parallel as well as sampling the solution, the files `decomposeParDict` and `sampleDict` are made and stored in this directory.

- The `constant` directory contains the `polyMesh` subdirectory where the files defining the mesh of the domain are stored, and other files specifying physical properties. In this study, the `transportProperties` file which defines the kinematic viscosity ν is included in the `constant` directory.

- The 'time' directory, usually starts with `0`, contains files for particular fields, e.g. velocity and pressure, at given times. In the `0` directory, the initial values and boundary conditions are specified. The subsequent time directories would usually contain results written to file by OpenFOAM, which are regulated by the settings in `controlDict`.

An additional directory would later include `postProcessing`, which contains the files written due to sampling the results.

4.3.2 Parallel run

As the domain of computation becomes more complex, the simulations need to be run on more than 1 CPU. In fact, throughout this study, every simulations are done on 80 CPUs, on the Serendipity server, owned by the Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente. Running a simulation on Serendipity was done through submitting 'jobs' via Slurm Workload Manager. The specific script for submitting the jobs is available upon request.

To specify the parallel simulations, the domain has to first be decomposed into partitions. This decomposition is controlled in `decomposeParDict` file located in the `system` directory, by the following command:

```
numberOfSubdomains  80;
method         scotch;
```

The simulations for parallel run is then executed using `mpirun` with a command line indicating the number of CPUs and the solver used, as follows:

```
mpirun -np 80 icoFoam -parallel > log 2>&1
```

4.3.3 The `icoFoam` solver

In accordance to approaching the modelling of the blood flow as an incompressible Newtonian fluid and considering the Reynolds number relevant to the flow in the Circle of Willis, the `icoFoam` solver was chosen to be the application solving the Navier-Stokes

equations. `icoFoam` is the "transient solver for incompressible, laminar flow of Newtonian fluids", using the PISO algorithm, requiring inputs of initial and boundary conditions [2].

The file containing the full code of the `icoFoam` solver and its supporting utilities can be found in OpenFOAM's tutorial directory (it can also be available upon request). The following contains brief explanation of the implementation of PISO algorithm in icoFOAM as well as the control files necessary to run the solver.

PISO algorithm in `icoFoam`

PISO (Pressure-Implicit with Splitting of Operators) algorithm was first introduced by Issa in 1986. The main feature of the algorithm is that rather than solves all of the coupled equations in a coupled or iterative sequential fashion, it partitions the operators into an implicit predictor and multiple explicit corrector steps.

An extensive explanation on the PISO algorithm and how it is implemented in `icoFoam` solver is provided in [2, 12, 20]. The next part will briefly discuss the essentials.

For strictly incompressible flow, the momentum and continuity equations are given by

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) = -\nabla P \tag{4.4}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{4.5}$$

where the coupling between density and pressure and between the energy equation and the rest of the system are removed. The non-linearity in the convection term $(\nabla \cdot (\mathbf{u}\mathbf{u}))$ is treated with an iterative solution technique $\nabla \cdot (\mathbf{u}\mathbf{u}) \approx \nabla \cdot (\mathbf{u}^0\mathbf{u}^n)$, where $\mathbf{u}^0$ is the current solution, $\mathbf{u}^n$ is the new solution, and the algorithm repeats until $\mathbf{u}^0 = \mathbf{u}^n$. While a constitutive pressure equation does not exist, the continuity equation imposes a scalar constraint on the momentum equation.

The idea of PISO is that while the pressure-velocity systems contain two complex coupling terms, on low Courant numbers (small time-step), the linear pressure-velocity coupling is much stronger than the non-linear **u-u** coupling of the convection term. Therefore, it is possible to repeat a number of pressure correctors without updating the discretization of the momentum equation (without updating $\mathbf{u}^0$). In such a setup, the first pressure corrector will create a conservative velocity field, while the second (and following) will establish the pressure distribution.

The advantage of PISO (in contrast to the Semi-Implicit Method for Pressure-Linked Equations (SIMPLE) algorithm) is that since multiple pressure correctors are used with a single momentum equation, it is not necessary to under-relax neither the pressure nor the velocity, which is usually needed to ensure stability of the computation but often slows down the convergence rate. On the other hand, the derivation of PISO is based on the assumption that the momentum discretization may be safely frozen through a series of

pressure correctors, which is true only at small time-steps. It is often experienced that the PISO algorithm is more sensitive to mesh quality than the SIMPLE algorithm [20].

An important aspect of CFD called the Rhie-Chow correction is often needed for flow simulations with collocated grid, which is the case in OpenFOAM simulations. When the pressure gradient does not depend on the pressure in adjacent cells, oscillations in the solutions occurs. The Rhie-Chow correction removes such oscillations [12]. An explicit equation for Rhie-Chow interpolation does not exist in OpenFOAM, but it is largely understood that OpenFOAM applies a correction 'in the spirit of Rhie-Chow'. The description of the approach is described in details in [12].

Numerical schemes in `icoFoam`

The numerical schemes used in icoFoam are defined in a file called `fvSchemes` located at the `system` directory. The `fvSchemes` file run for this study can be available upon request. The terms required to be assigned for `icoFoam` are as follows:

- `ddtSchemes`, which refers to the time derivative $\frac{\partial}{\partial t}$. The discretisation scheme for this term is the `Euler` scheme, which is a first order, bounded, implicit discretisation.
- `gradSchemes`, which refers to gradient (∇) terms. The `Gauss linear` discretisation is chosen for both the `default` and `grad(p)` (∇P) entries, which is a linear, second order, Gaussian integration scheme.
- `divSchemes`, referring to divergence ($\nabla \cdot$) terms. The only discretisation available for the convection term `div(phi,U)` is the `Gauss` scheme with `linear` interpolation, which gives a second order, unbounded numerical behaviour.
- `laplacianSchemes`, which contains the Laplacian (∇^2) terms . The `Gauss` scheme is the only discretisation available for this term, with specification following `linear` interpolation and `orthogonal` surface normal gradient scheme.
- `interpolationSchemes`, referring to point-to-point interpolations of values. The `default` entry is assigned to be `linear`, which uses central differencing.
- `snGradSchemes`, which refers to component of gradient normal to a cell face. A surface normal gradient is evaluated at a cell face; it is the component, normal to the face, of the gradient of values at the centres of the 2 cells that the face connects. A surface normal gradient may be specified in its own right and is also required to evaluate a Laplacian term using Gaussian integration. The basis of the gradient calculation at a face is to subtract the value at the cell centre on one side of the face from the value in the centre on the other side and divide by the distance. The calculation is second-order accurate for the gradient `normal to the face` if the vector connecting the cell centres is orthogonal to the face, i.e. they are at right-angles. This is the `orthogonal` scheme which is given as the `default` entry.

Solution and algorithm control

The file `fvSolution`, located in the `system` directory, contains entries that control the equation solvers, tolerances, and algorithms. Each `fvSolution` file is specific to the solver being run, and the file for `icoFoam` can be found in Appendix ?. Brief descriptions of the entries are as follows:

- `solvers` control, with entries for `U` and `p`, since `icoFoam` solves equations for velocity **u** and pressure P.
  - For pressure, the `solver` is `PCG` (Preconditioned conjugate gradient), with `preconditioner` set to `DIC` (Diagonal incomplete-Cholesky).
  - For velocity, the `solver` is `smoothSolver` with `smoother` set to `symGaussSeidel` which refers to symmetric Gauss-Seidel method.
  - The solver tolerances control the stopping criteria for solver iteration. In `icoFoam`, the solver stops when either the residual falls below the `tolerance` or the ratio of current to initial residuals (i.e., a measure of the error in the solution) falls below the `relTol` (relative tolerance). The `tolerance` refers to the level at which the residual is small enough that the solution can be accepted as sufficiently accurate, while the `relTol` restrains the relative improvement from initial to final solution. There is another optional entry of maximum iteration `maxIter`, which when not specified takes the default value of 1000. For this study, the solver setting for pressure is as follows.

```
p
{
    solver          PCG;
    preconditioner  DIC;
    tolerance       1e-06;
    relTol          0.05;
}
pFinal
{
    $p;
    relTol          0;
}
```

    This setting takes into account that in the PISO algorithm the pressure equation is solved multiple times. In the case where within one time step the pressure is solved 3 times, the first two solutions uses the setting for `p` with `relTol` of 0.05 as to set the cost of computation to be relatively low. The final solution

will take the setting `pFinal`, which deactivate `relTol` (by setting it to 0) and the equation is solved to a residual level specified by `tolerance` for better accuracy but at a greater computational cost.

- `PISO` control, which specifies how the PISO algorithm loops over the equations.
  - The `nCorrectors` entry, which sets the number of times the algorithm solves the pressure equation and momentum corrector in each step. The default value is 2 and usually not more than 4.
  - The `nNonOrthogonalCorrectors` entry, which specifies repeated solutions of the pressure equation, used to update the explicit non-orthogonal correction of the Laplacian term. In this study it is set to 0, since we use orthogonal mesh.
  - The `pRefCell` and `pRefValue` entries are both set to 0 by default. These entries refer to pressure referencing for which we have not develop an intuition yet.

Initial and boundary conditions

The initial and boundary conditions of each field are defined in files stored at the starting time in directory `0`. For this study, the two fields involved are the velocity and pressure defined by the files `U` and `p` respectively.

For the velocity field, the case of steady-state flow uses the following setting:

```
dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
    walls
    {
        type            noSlip;
    }

    inlet
    {
        type            fixedValue;
        value           uniform (0.1017099057 0 0);
    }

    outlet
    {
```

```
        type            zeroGradient;
    }
 }
```

For the pressure field, the setting is as follows:

```
dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    walls
    {
        type            zeroGradient;
    }

    inlet
    {
        type            zeroGradient;
    }

    outlet
    {
        type            fixedValue;
        value           uniform 0;
    }
}
```

The explanation on those entries is as follows:

- The `dimensions` entry defines the dimension of the field, which in this case is m/s for velocity and m^2s^{-2} for the *kinematic* pressure.
- The `internalField` defines the value of internal field, which is initialised as uniform zero in three vector components for velocity since velocity is a vector, and as uniform 0 for pressure since in an incompressible case its absolute value is not relevant.
- The `boundaryField` includes the boundary conditions (BC) and necessary data for the boundary patches.

– The `zeroGradient` BC is applied to the `wall` and `inlet` for pressure and to the `outlet` for velocity. This is the Neumann boundary condition, which prescribes that the value at the boundary takes the value at the grids just before the boundary. In chapter 6 this boundary condition choice will be clarified further.

– For the velocity at the `wall` the BC applied is the `noSlip` BC. This is the Dirichlet boundary condition which in this case means the velocity at the wall is set to 0.

– The `fixedValue` condition is applied at the `outlet` for pressure and at the `inlet` for velocity. The uniform 0 value of pressure at the `outlet` has the same effect as the uniform value 0 at its internal field. The uniform value of `(0.1017099057 0 0)` set at the `inlet` for the velocity indicates that the initial velocity moves at approximately 0.1 m/s in the x-direction. This value depends on the chosen Reynolds number Re, which will be elaborated more in chapter 6. Additionally, a different type of condition will later be prescribed when solving for a pulsating flow (section 6.3)

Chapter 5

Validation of the simulation method

While the implementation of the Immersed Boundary Method (IBM) in OpenFOAM is relatively new, there have been several studies aimed to validate and verify the use of the method on this platform. A study by Constant et al. in 2017 [10], in particular, presented a verification and validation for IBM in which the implementation of the method is done through modifying the PISO algorithm used in the solver `pisoFoam`. The study applied the modified solver in various 2D and 3D simulations for Reynolds numbers ranging from 30 to 3900, i.e., from 2D steady to turbulent regimes, and compared the result to existing data from literature and OpenFOAM native models. The analysis of the computational cost, numerical behavior and accuracy of the numerical method in the study showed that the global properties of the OpenFOAM solver are not altered by IBM and hence the study validated the method.

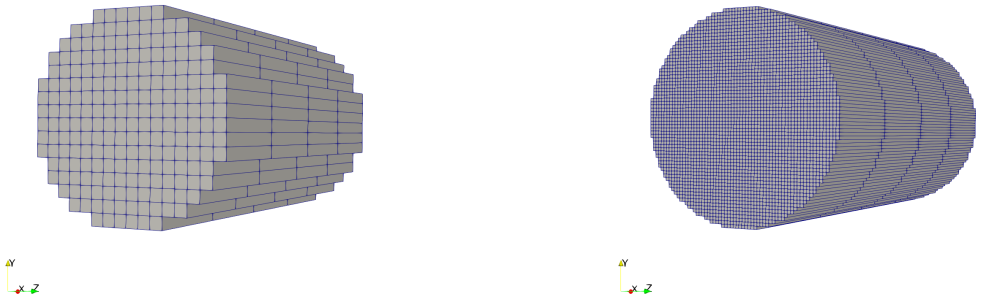
In this study, however, the implementation of IBM is not done by modifying the solver. It is instead implemented during the mesh generation which converted the `.stl` file into a staircase geometry representation ready for OpenFOAM simulations. This step is done using an in-house code which allows to generate the IB masking function from a boxed `.stl` representation. The geometry represented in `.stl` format is first transformed into a geometry on a 3D grid. The IB masking function is determined by scanning over all grid cells inside the transformed geometry and asking whether a grid cell is of type 'fluid' or 'solid'. The 3D grid geometry resulting from this is the masking function that can be used further in the Immersed Boundary method. Once this geometry is defined, a set of files is created to accommodate the format required by OpenFOAM, by-passing the representation usually created using `blockMeshDict`. In short, the implementation of IBM in this study is done while defining the mesh on the domain of computation.

Since the implementation of IBM in OpenFOAM for this study differs from the ones found in literature, a brief validation was done for a simple case, before tackling more complex geometries. The rest of this chapter will describe this validation process.

The laminar flow in a straight pipe has a known analytical solution, known as the Poiseuille's flow. The Poiseuille's flow is a parabolic flow profile where the maximum

velocity occurs in the middle of the pipe and steadily decreases to 0 as it nears the wall of the pipe. This is the basis of a validation of IBM on OpenFOAM in which the simulations are done on a straight pipe with various grid resolutions, with the purpose of confirming that the numerical solutions converge to the analytical Poiseuille's flow solution.

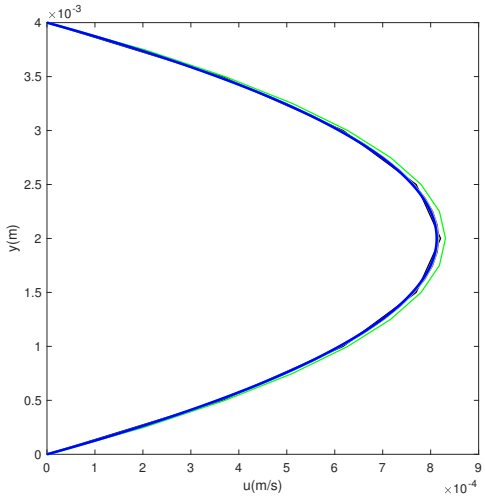
The domain of the simulations was a pipe with diameter 4mm and length 8 mm. For simplification, the Reynolds number chosen was $Re = 1$, which corresponds to a maximum velocity of 8.125×10^{-4} m/s. For grid refinement analysis, the geometry was set up from 8 by 8 grid-cells (2 grid-cells per mm), increased by the factor of 2 up to 256 by 256 grid-cells (64 grid-cells per mm). Figures 5.1(a) and 5.1(b) show the IBM geometry with 16 by 16 grid and 64 by 64 grid respectively, where the pipe becomes smoother for finer grids.



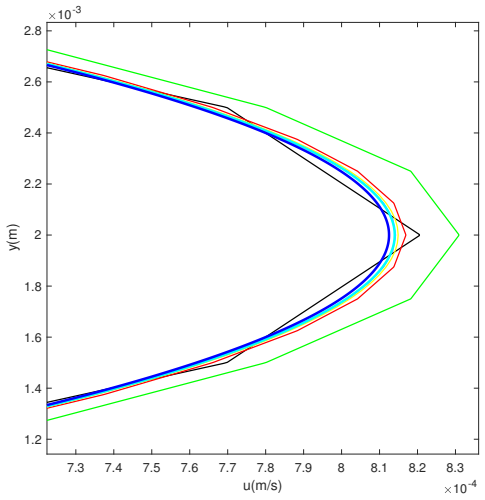
(a) (b)

Figure 5.1: (a) IBM pipe, 16 grid-cells,
(b) IBM pipe, 64 grid-cells.

The resulting velocity profile is shown in fig. 5.2. The OpenFOAM-based IBM produced Poiseuille's flow profile, quite independent of the spatial resolution. Looking closer at the maximum of the graphs, as seen in fig. 5.2(b), we conclude that the numerical solutions converge to the analytical solution (the blue graph). We also looked more closely into the convergence behavior and confirmed second order accuracy of the simulation method by considering the convergence rate of the maximal velocity as function of the grid refinement (see also [21]).



(a)



(b)

Figure 5.2: Parabolic velocity profile for different grid resolutions - simulations at Reynolds number $Re = 1$ are included at resolutions ranging from 2 grid cells per mm to 64 grid cells per mm in a pipe geometry of diameter 4 mm and length 8 mm. In (a) the total view is shown and in (b) a zoomed version near the center of the channel.

Chapter 6

Flow diversion prediction on a model geometry for a side-wall aneurysm

Having presented the geometry handling process and the numerical methods employed, as well as having validated the method of IBM in OpenFOAM, we can now describe the process and results of simulations done on a model geometry of a vessel with a side-wall aneurysm. This chapter will present detailed information about the model geometry (Section 6.1), describe the setup and the qualitative and quantitative analysis of a steady flow (Section 6.2), and discuss the effect of pulsatile flow in the model geometry (Section 6.3). Changes in the flow due to the inclusion of a particular stent will be highlighted.

6.1 The model geometry

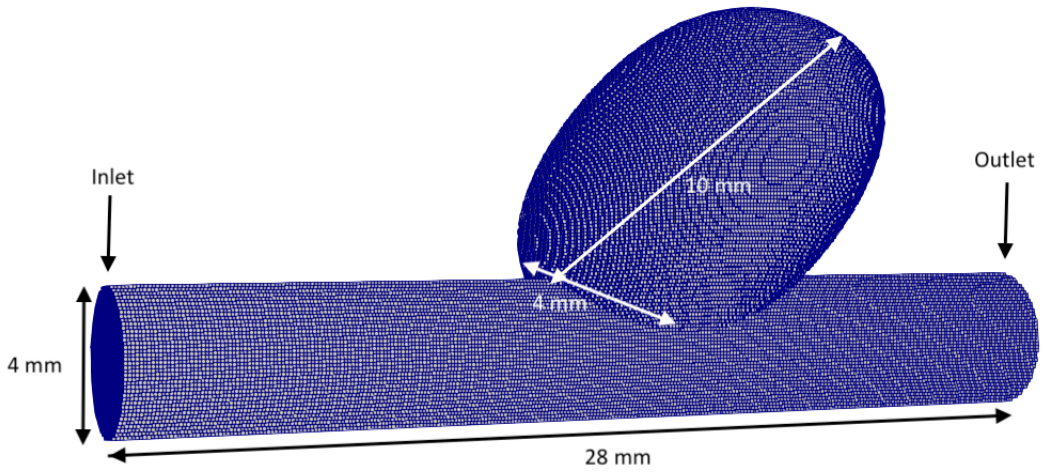


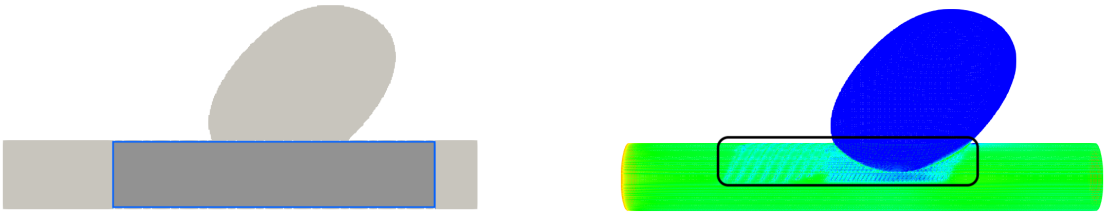
Figure 6.1: The basic geometry of vessel with aneurysm.

The model of a vessel with side-wall aneurysm is illustrated in fig. 6.1, which is fig. 3.1(a) that has undergone the process of IBM mesh generation. The length of the vessel is 28 mm, with diameter 4mm. The length of the neck (opening) of the aneurysm is

4mm, while the height of its dome is 10 mm. The size of the aneurysm, with dome-to-neck ratio of 2.5, is the typical size of aneurysms usually treated using flow diverter stenting; the coiling treatment is often unsuccessful due to the large opening causing the coil to escape back into the vessel.

The two implants used in this study were an artificial flow diverter (FD) constructed based on the typical flow diverter on the market and the eCLIPs. The thickness of the wires of the FD is 0.03 mm, and the wires are braided with density approximately 6 wires per mm. The length of the FD is 20 mm and the diameter is adjusted to the diameter of the vessel which is 4mm. The eCLIPs is non-circumferential, with a 'rib-like' structure that has the thickness of 0.06 mm attached to a 'spine' of thickness 0.1 mm. The anchoring segment of the geometry is 7mm long and less dense at 2 'ribs' per mm, while the flow-diverting segment of the eCLIPs is 8mm long and more dense at 5 'ribs' per mm.

Figures 6.2(a) and 6.2(b) show the location of the FD and the eCLIPs respectively. The FD is located approximately 6.5 mm away from the inlet and so the neck of the aneurysm is covered fully by the FD, and the length of the FD structure before and after the aneurysm is similar. The eCLIPs is placed approximately 7mm away from the inlet, ensuring that the flow-diverting segment fully covers the opening of the aneurysm.



(a) (b)

Figure 6.2: The location of (a) FD (darker gray), and (b) eCLIPs.

The simulations were done for the geometry with 10, 15, 20, and 30 grid-cells per mm, with details as seen in table 6.1.

Grid-cells per mm	Total Grid-cells
10	$280 \times 130 \times 110 = 4004000$
15	$420 \times 195 \times 165 = 13513500$
20	$560 \times 260 \times 220 = 32032000$
30	$840 \times 390 \times 330 = 108108000$

Table 6.1: The grid-cells resolutions of the domain.

The number of grid-cells used in these simulations, admittedly is less than ideal, especially in resolving the detail of the FD's wires. Since the thickness of the FD's wires is 0.03 mm, only the 30 grid-cells resolution actually captures some of the detail of the

wires. However, as will be seen in later sections, this has less effect on the overall flow inside the whole domain.

6.2 Case: steady-state flow

The simulations on OpenFOAM were set up to solve the incompressible Navier-Stokes equations, as laid out in eq. (4.1). At the wall, the Dirichlet boundary condition, i.e., no-slip condition, was imposed for the velocity. At the outlet, we imposed the Neumann boundary condition, i.e., zero-gradient condition, for the velocity. Zero-gradient condition means the velocity at the outlet took the value of the velocity at the grid cells just before the outlet, as to simulate the continuing developed flow that happens in the actual long vessel in reality. A uniform velocity was imposed at the inlet, taking into account that after a certain entry length, the velocity would develop into the desired Poiseuille type flow, as illustrated in fig. 6.3.

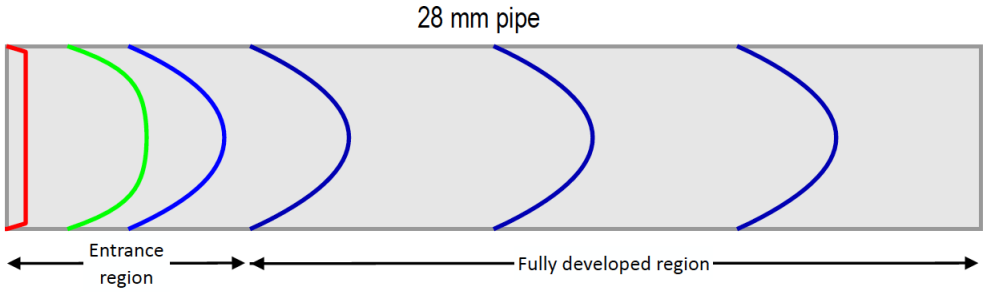


Figure 6.3: Development of flow into Poiseuille flow from uniform velocity imposed at the inlet.

The value taken as the uniform velocity imposed at the inlet was given dependent on the Reynolds number Re, a dimensionless quantity which describes the flow regime. In this study, the value used is $Re = 250$, which is within the range of Re for blood flow in the human brain [15, 16]. The Reynolds number Re is given by:

$$Re = \frac{\mathbf{u}L}{\nu}, \tag{6.1}$$

where L is the characteristic length which for pipe-like structures is taken to be the diameter of the pipe $L = 4\text{mm}$, ν is the kinematic viscosity of the blood taking the value $\nu = 3.25 \times 10^{-6}\text{m}^2/\text{s}$, and $\mathbf{u}$ is the characteristic velocity. From eq. (6.1), it can be seen that the characteristic velocity $\mathbf{u}$ is given by:

$$\mathbf{u} = \frac{\nu Re}{L}. \tag{6.2}$$

For $Re = 250$, the maximum velocity is $\mathbf{u}_{\text{max}} = 0.2034198113\text{m}/\text{s}$, which gives the average velocity $\mathbf{u}_{\text{avg}} = 0.1017099057\text{m}/\text{s}$. This average velocity was the value of the uniform

velocity imposed at the inlet (recall section 4.3).

For pressure, we imposed zero-gradient conditions at both the wall and inlet, while at the outlet we imposed a uniform value of zero (0).

6.2.1 Qualitative analysis

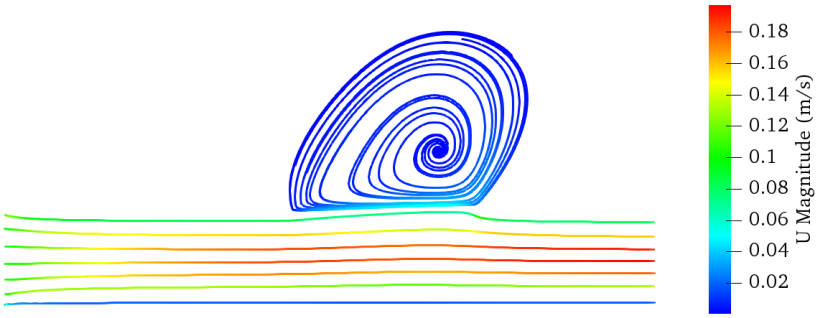


Figure 6.4: Velocity profile for geometry without implants.

Figure 6.4 shows the velocity profile for the geometry without implants for 30 grid-cells per mm as a stream representation. Inside the vessel, a formation of the Poiseuille flow can be observed, where the highest velocity represented by the red lines is located in the middle of the vessel, which then decreases gradually as it gets closer to the wall where the lines become blue. This formation of a parabolic flow profile occurs after a certain entry length. Inside the aneurysm sac, the flow appeared to be very low throughout - in this representation only the colour blue can be seen in the sac. Hence, we set a log-scale graph, as seen in fig. 6.5, which gives a better impression of the details of the flow inside of the aneurysm.

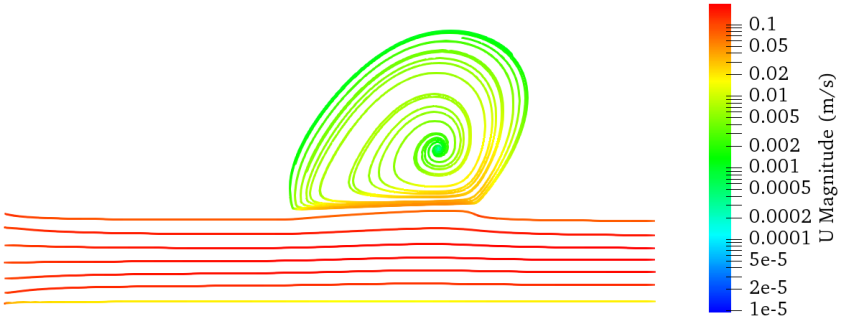


Figure 6.5: Log-scale velocity profile for geometry without implants.

The flow entering the aneurysm sac was at its highest at the point further away from the inlet, which indicates the location in the sac receiving the most impact from the water hammer effect. Additionally, the flow was formed into a structure with a vortex located slightly to the left part.

Figure 6.6 depicts the velocity profile for the geometry with implants installed in log-scale. It can be seen from the graphs that both the FD and the eCLIPs reduced the flow

entering the aneurysm sac, as well as disrupt the forming of the vortex. At the first glance, it appears that there is no striking difference in the overall effect of installing either of the implants.

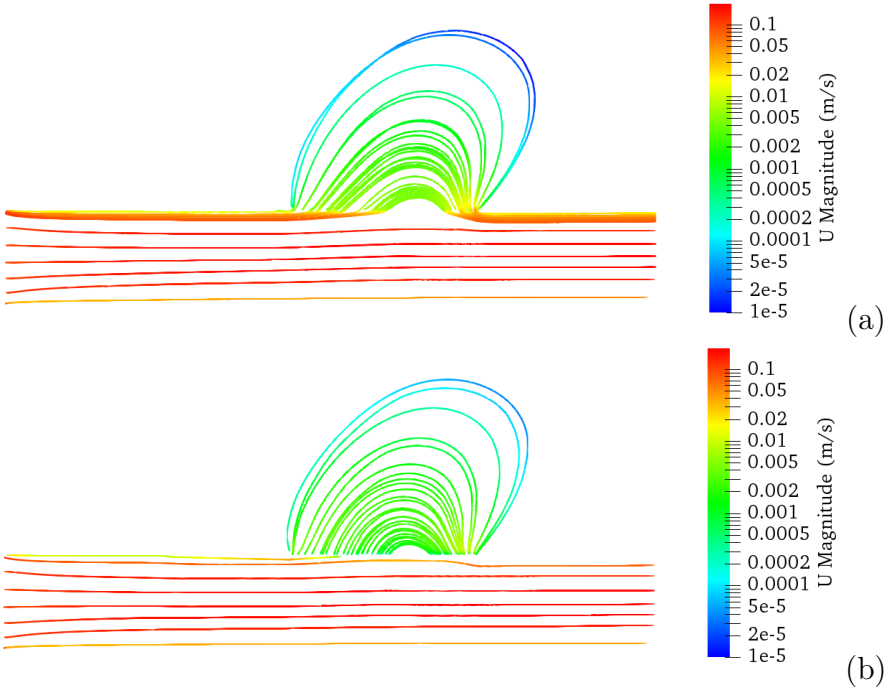


Figure 6.6: Log-scale velocity profile for geometry with implants: (a) with FD, (b) with eCLIPs.

Looking closer into the immediate surrounding of the wires or metal part of the FD, we can also see that there was a significant decrease of the flow as it moved from the vessel into the aneurysm sac (fig. 6.7). As will be evident in the next section as well, while not affecting the overall flow impression, the available resolution has some effect in the minute details of the flow very near the wires of the implant. Further away from the implant the effects of the adopted resolution are much less visible.

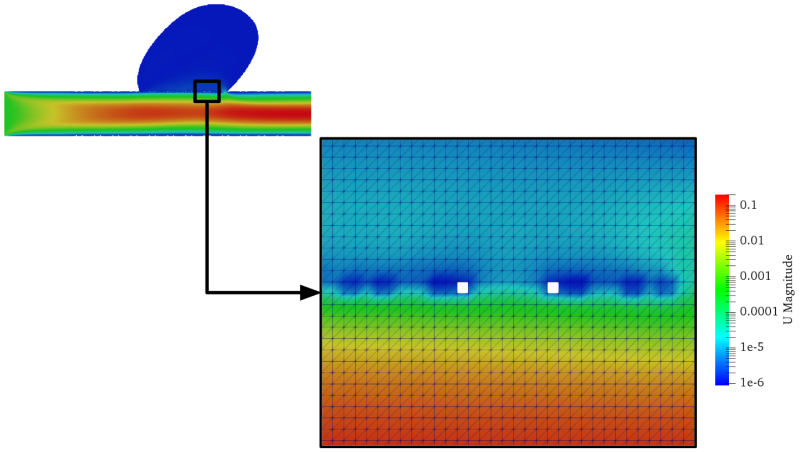


Figure 6.7: Flow at the immediate surrounding of the wire of the FD.

6.2.2 Quantitative analysis

For quantitative analysis, the solutions are sampled along 5 lines, in the geometry with and without implants. The lines sampled are line 1 at $x = 0.0028$m which is 10% of the total length of the vessel, line 2 at $x = 0.0085$ which is approximately 10% of the implants, line 3 at $x = 0.01415$m which is around a quarter into the aneurysm, line 4 at $x = 0.0182$m, which is around three quarter into the aneurysm, and line 5 at $x = 0.0228$m which is approximately 10% of the total length of the vessel after the aneurysm.

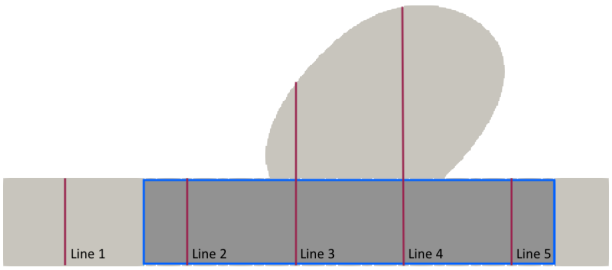


Figure 6.8: Vertical lines of sampled solutions.

In fig. 6.9, the position $y = 0$ to $y = 0.004$ represents the vessel, while the position $y > 0.004$ indicates the aneurysm sac. The **black** lines represent the flow on the geometry without implants, the blue lines indicates the flow affected by the FD, while the red lines show the flow when eCLIPs is installed. Confirming the qualitative impression, the velocity profiles for lines 3 and 4, as depicted in fig. 6.9 show that both the FD and the eCLIPs reduced the flow into the aneurysm, with eCLIPs appearing to be only marginally better at blocking the flow.

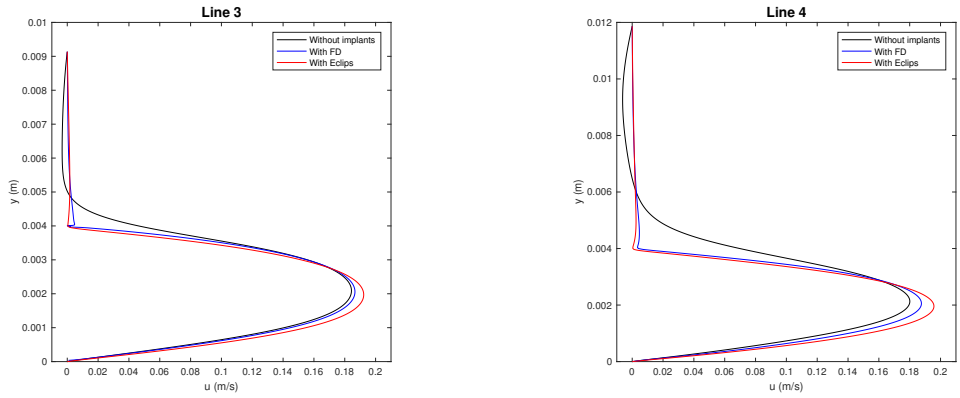


Figure 6.9: Velocity profile for sampled lines 3 and 4, finest grid-cell resolutions.

Looking at the profile of the flow at the position $y = 0$ to $y = 0.004$, it can be observed that installing the implants increases the velocity in the vessel. The eCLIPs in particular, produces an increase of approximately 8.7%, which is more than twice as much as the increase of 4.2% caused by the FD. This effect can be attributed to the non-circumferential structure of the eCLIPs, which produces less friction when installed in the vessel, compared to the cylindrical structure of the FD.

To check for convergence, the solution along Line 4 are graphed for all grid resolutions. The **black** line represents 10 grid-cells per mm, the red line 15 grid-cells per mm, the green line 20 grid-cells per mm, and the blue line 30 grid-cells per mm. In both the geometry without implants and with eCLIPs, the solutions appear to be independent of the grid-cell resolutions; for the geometry without implants the graphs overlap, for the geometry with eCLIPs the graphs almost coincide. This indicates that the solutions are very reliable.

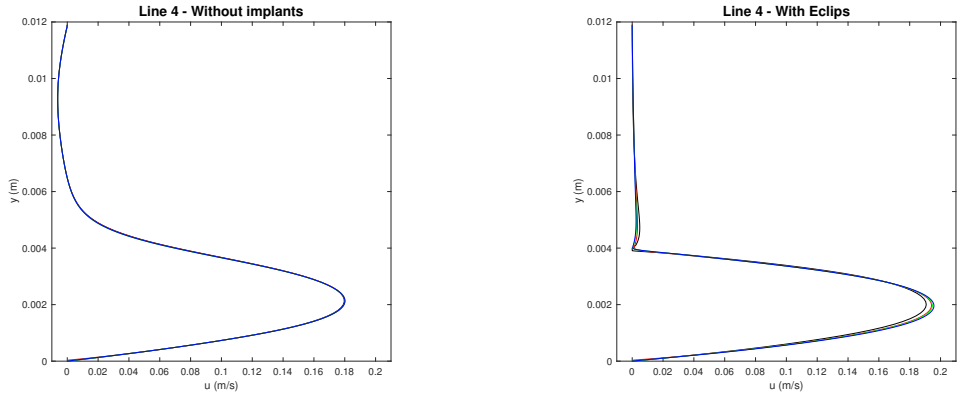


Figure 6.10: Velocity profile for sampled line 4 in geometry without implants and with eCLIPs, all grid-cell resolutions.

As the structure of the FD is more finely detailed, the grid-cell resolutions do affect the solutions for the geometry with FD. This is the most apparent in the structure of the flow where it is entering the aneurysm (at $0.004 < y < 0.006$) as seen in fig. 6.11. However, a convergence of the solution is still observed and it is expected that refining the grid resolutions further, while beneficial to capture the more detailed nuance of the flow near the thin wires, will not alter the overall flow impression.

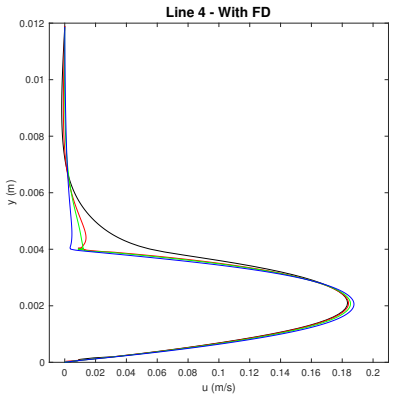


Figure 6.11: Velocity profile for sampled line 4 in geometry with FD, all grid-cell resolutions.

The analysis of the steady flow in the model geometry appears to suggest that while eCLIPs produces a slightly better flow blocking effect, both implants are equally effective at diverting the flow away from the aneurysm. In the event where reduction of friction in the vessel is desired, then the eCLIPs is the device that produces less friction.

6.3 Case: pulsatile flow

While analyzing the steady case is needed to get the overall impression of the flow, to move further into approximating real cases, the next step to take is investigating the effect of a pulsatile variation of the flow. This section will describe the set up for the simulation of the pulsating flow and discuss the observation of the results.

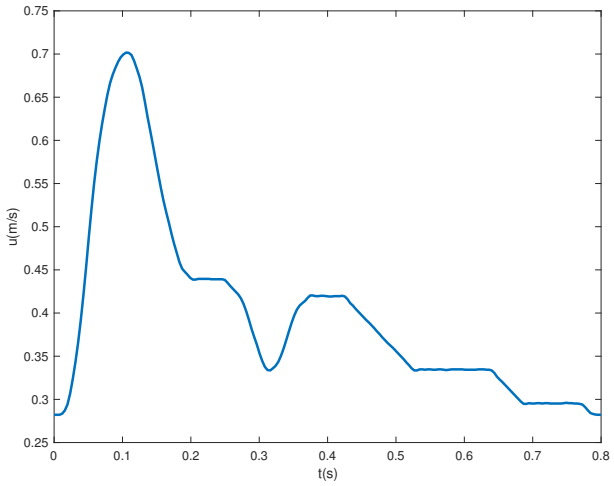


Figure 6.12: A typical pulse signal chosen to impose the pulsating variation of the flow.

A typical pulse as depicted in fig. 6.12, was chosen, taking reference from [15], where the average velocity of the blood flow is rounded off to 0.4 m/s, and one cardiac cycle lasts for the physical duration of 0.8 s. In OpenFOAM, this typical pulse is then implemented at the inlet of the latest time directory of the steady-state cases (2s). Starting with the steady case was proven necessary to ensure that the simulations can run without having to develop the flow from the beginning. Recalling back to Section 4.3, the `inlet` setting of the velocity field `U` is changed to

```
    inlet
{
    type            uniformFixedValue;
    uniformValue    tableFile;
    uniformValueCoeffs
    {
        outOfBounds     repeat;
        file            "pulse_08_40";
    }
    value           uniform (0.4402533250000001 0 0);
}
```

where the file `"pulse_08_40"` contains the pulse signal and the time corresponding to the signal.

The simulations were then run from 2s to 5s, where the resulting pulse signal recorded in Line 4 of the sampled solutions is as depicted in fig. 6.13. This recorded pulse has an average velocity that is slightly faster than 0.4 m/s. The first full periodic pulse started at 2.41s, the second at 3.21s, and the third at 4.01s. Taking the second full periodic pulse, we have corresponding to 4 points of interest during a cardiac cycle, peak systole at $t_1 = 3.21$s, end systole at $t_2 = 3.31$s, peak diastole at $t_3 = 3.42$s, and end diastole at $t_4 = 3.75$s.

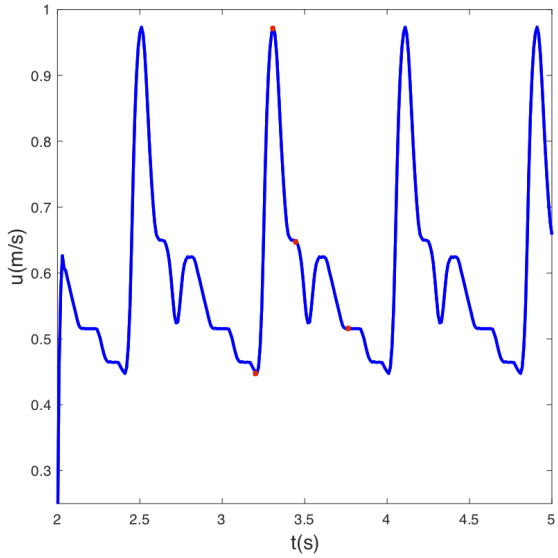


Figure 6.13: The recorded pulse showing three fully periodic pulses. The red dots at the 2nd periodic pulse indicate the 4 points of interest during a cardiac cycle: peak systole, end systole, peak diastole, and end diastole.

The snapshots of the flow at the four points of interest in the geometry without implants can be seen in fig. 6.14. It can be clearly seen at end systole ($t_2 = 3.31$) that the flow enters the aneurysm sac at a relatively high velocity, which did not happen in the steady-state case. The water hammer effect is more prevalent under the pulsatile flow condition.

Comparing the effect of the implants, it can be seen in fig. 6.15 that both implants reduce the flow entering the aneurysm sac, even during end systole. In this case, eCLIPs appears to be more effective at flow-diverting. However, since the pulsating simulations were done on the coarsest grid-cell resolutions (10 grid-cells per mm), the significant difference in flow-diverting effectiveness could be caused by the low grid-cell resolution which affects the FD more than the eCLIPs.

Looking closer at the solutions sampled at Line 4 from fig. 6.8, the flow with pulsatile variations no longer form a parabolic profile. Furthermore, in fig. 6.16(a) it can be seen that there are more flow variation happening inside the aneurysm, most prominently at peak diastole. Figures 6.16(b) and 6.16(c) show once again that both implants reduce the flow entering the aneurysm sac, but with pulsatile variation, there are some flow movements

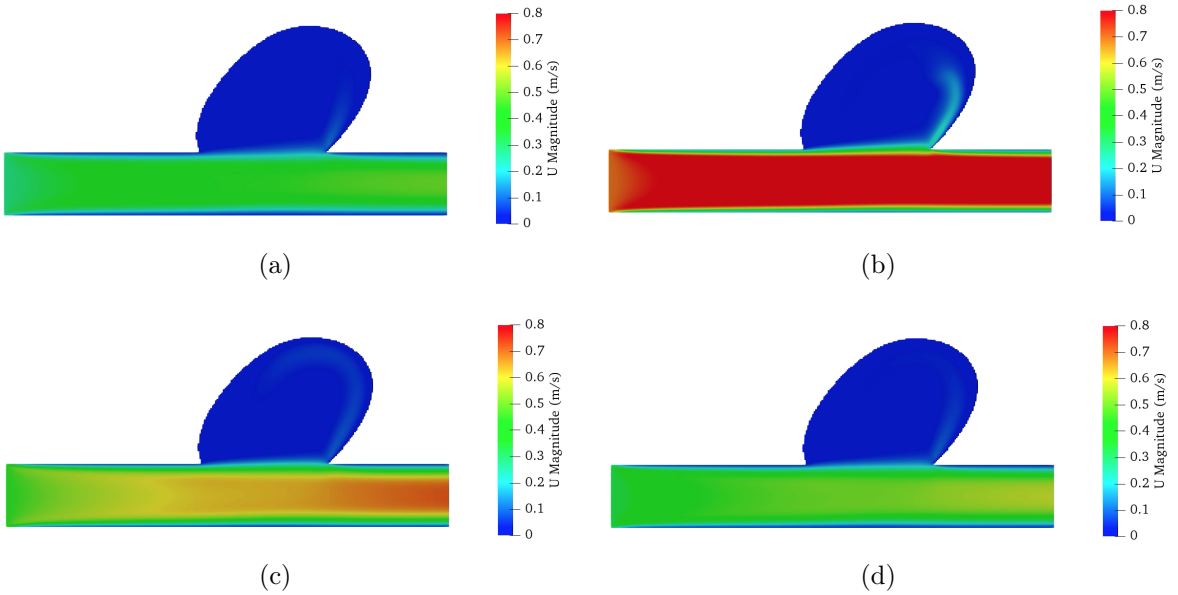


Figure 6.14: Snapshots of the flow in the geometry without implants during pulsatile variation at the four point of interest: (a) peak systole ($t_1 = 3.21$s), (b) end systole ($t_2 = 3.31$s), (c) peak diastole ($t_3 = 3.42$s), and (d) end diastole ($t_4 = 3.75$s).

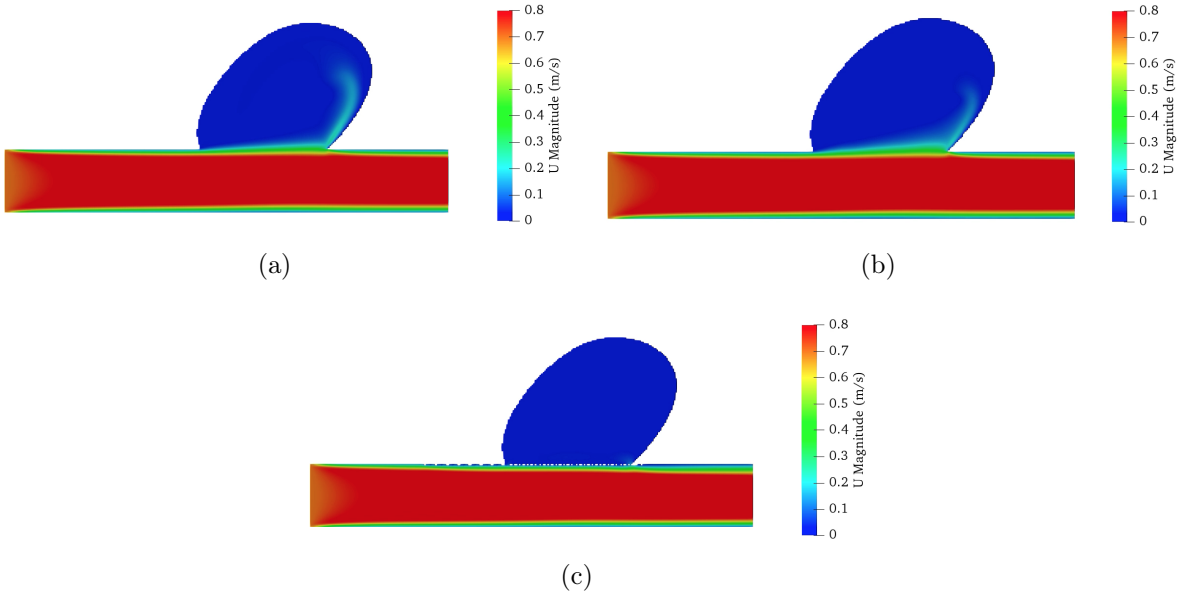
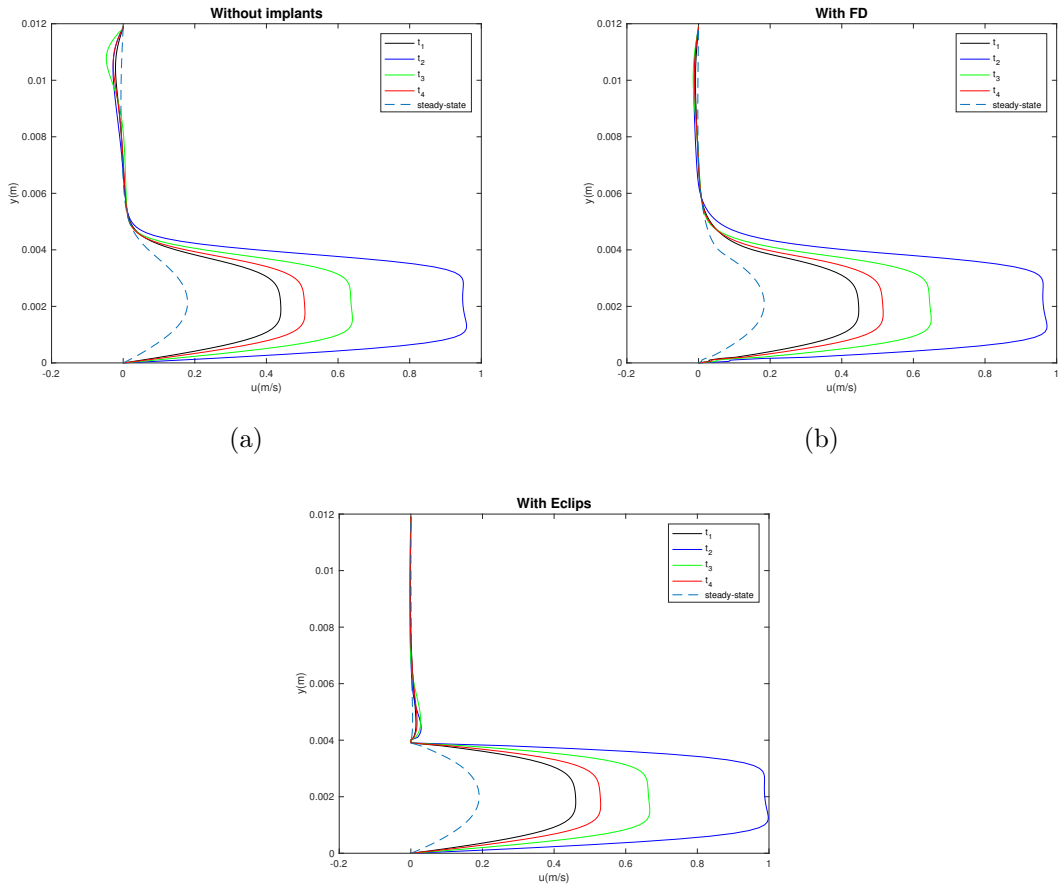


Figure 6.15: Snapshots of the flow at end systole ($t_1 = 3.31$s) for the geometry (a) without implants, (b) with FD, and (c) with eCLIPs.

at the point right at the start of the aneurysm (around the position $y = 0.004$).

Focusing on end systole, fig. 6.17 depicts the comparison of the flow-diverting effect by the two implants, showing the eCLIPs to be more effective than the FD. Additionally, as similarly seen in the steady-state case, eCLIPs is also better at reducing friction inside the vessel, as evident by the higher velocity represented by the red line in fig. 6.17.

Under the pulsatile condition, where the pulse is close to the physiological condition



(a) (b)

(c)

Figure 6.16: Sampled solutions at Line 4, at the four points of interest for geometry (a) without implants, (b) with FD, and (c) with eCLIPs.

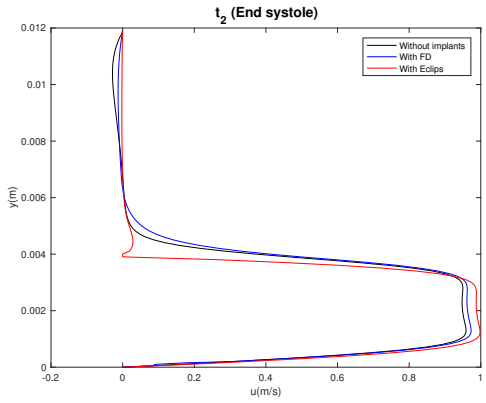


Figure 6.17: Sampled solutions at Line 4 at end systole showing the flow comparison between the geometries with and without implants.

of an average person at rest, both the FD and eCLIPs are shown to divert the flow from entering the aneurysm. Pending simulations with better grid-cells resolutions, the eCLIPs is also shown to be more effective both at diverting the flow away from the aneurysm sac and at reducing the friction in the vessel.

Chapter 7

Conclusions and Outlook

In this report we laid the foundations toward a systematic and personalized analysis of the flow in stented aneurysms, connecting, on the one hand, the patient's recorded vessel geometry near the diseased area to, on the other hand, the corresponding prediction of the flow before and after the application of a particular flow diverting stent. The entire workflow was illustrated for a mathematical model geometry. The main findings and required steps will be described in this chapter. Subsequently, we will discuss some recommendations for future research toward the remaining main challenge, i.e., achieving such predictions in actual patient geometries. Results of preliminary simulations in that direction, illustrating the potential of the approach will be shown.

7.1 Toward an automated flow analysis of stented aneurysms

Building a reliable prediction of the flow in a complicated domain requires mastering a number of steps. We discuss the main steps next.

- A first, basic step involves the *definition of the flow domain* in all details. The starting point for such a representation is the recorded patient geometry that is available in DICOM format. Often a scan of the entire skull is recorded of the patient. Via precise segmentation it is possible to extract the relevant structures in the required sub-volume in which the affected aneurysm and local vasculature are contained. From the many recorded grey scales in the original DICOM format, only 'inside' and 'outside' the flow domain are relevant for the adopted IB method - via segmentation this can be achieved. The step from DICOM to the `.stl` format that represents the segmented geometry was considered using the HOROS free software tool. Alternative, more powerful software is available for such task, but for current developments are not yet required. Possibly, when further automation of the entire workflow is considered, alternative professional software should be included.

- An important next step in the analysis involves the *preparation, simplification and smoothing of the raw geometry*. The raw data contain a sometimes rather high level of noise in the recorded imagery. Moreover, when flow in an aneurysm is required, many of the vessels in the general vicinity do not have a strong influence on the flow details inside the aneurysm sac. Using software that can manipulate `.stl` files in terms of smoothing of the noise and simplification by algorithmically removing the many smaller vessels, one can prepare the geometry for the subsequent flow analysis. Such prepared geometries are of course (close) approximations of the actual situation – based on earlier investigations ([15]) it appears, however, that the sensitivity of the predicted flow to small-scale deviations in the vicinity of the aneurysm is not too large. Hence, smoothing and simplification are robust steps in the total workflow. We used BLENDER for this task.

- A third step is the *incorporation of a candidate flow-diverting stent* in the diseased situation. This process is also known as 'virtual stenting' and requires the merging of a detailed model of the stent with the local diseased vessel shape. This is a step that currently requires considerable manual operations, leading to a close integration of the generic stent model with the particular flow domain found in a patient. Also in this instance BLENDER is a helpful tool, allowing to manipulate `.stl` files such that the final, highly complex geometry with stent are in place inside the recorded vessel. Currently, deformations of the local vessels as a result of placing a somewhat stiff device are not part of the analysis - this aspect of the 'virtual stenting' problem deserves more attention in the future - in this study we assumed the recorded vessel geometry to be rigid.

- A fourth step is the *translation of the `.stl` format of the total prepared flow domain to an IBM representation* suitable for flow simulations with OpenFOAM. This step requires embedding the geometry as represented and manipulated in BLENDER in a rectangular volume, within which the IB masking function can be determined by scanning over all grid cells inside the embedding rectangle and asking whether a grid cell is of type 'fluid' or 'solid'. Since the outermost grid cells of the embedding rectangle are either inflow/outflow or 'solid', such assignment of type 'fluid' or 'solid' can be done systematically, looping over the grid.

- A fifth and major step from the perspective of computations is the *simulation of the (pulsatile) flow* that arises. For this purpose, we rely on OpenFOAM, although other software platforms could take this place as well. We find OpenFOAM suitable for this task - as the flows of interest take place at low Reynolds numbers, the sensitivity of the outcome to the smallest details of the domain is modest. This implies that we may achieve accurate predictions at feasible spatial resolutions. This does not mean that the simulations are 'cheap' - in fact, when the need arises that the flow

should be resolved on the scale of the individual wires of a flow-diverting stent, then this sets a rather strict requirement on the spatial resolution, e.g., adopting at least 2 grid cells across a cross-section of a wire in the stent. This can be a very high resolution, despite the otherwise laminar pulsatile flow. Conversely, as such very small length-scales do not arise in the natural, un-stented geometry, resolution requirements for realistic aneurysms are much lower.

- A sixth, final step involves the *visualization and analysis* of the computed results. A mix of visualization tools such as ParaView and other software such as Matlab was adopted. This allows 3D flow representation, the analysis of profiles of flow properties at interesting parts of the domain (e.g., planes, lines) and the comparison of the predictions at different resolutions and with different stents.

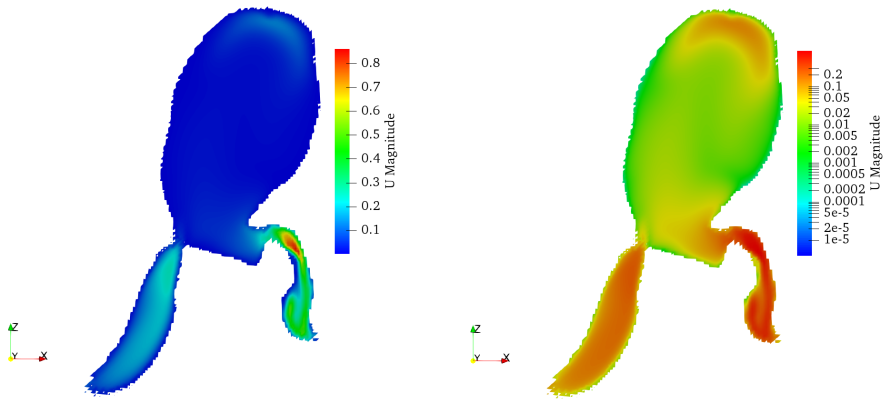
In this report we described the major elements of the workflow in detail and could illustrate all steps for the mathematical benchmark domain that was specifically constructed for this. This is the main result of the work - it lays the foundation for more relevant simulations of actual patients. The analysis of such realistic cases is in principle comparable to that of the model geometry. However, some specific challenges remain, as we will discuss next.

7.2 Challenges and achievements for realistic geometries

Based on our experience with the model geometry, we could successfully master the problems of (a) flow domain representation, (b) virtual stenting, (c) flow simulation and (d) flow analysis. As reported, both steady and pulsatile flow was handled in the simple mathematical geometry, either without or with a stent in place. To render this capability of some clinical relevance it is essential to be able to extend these steps to actual real geometries. In the course of the work we managed several important sub-steps in the process. We discuss this below and also highlight the main problems that still need attention and research.

- The first problem we could handle in more general, realistic geometries, is that of *steady flow in actual patient-specific aneurysms*. We could work with two patient geometries, one we refer to as the Portugal geometry, discussed in [11] and one we refer to as Nijmegen-1, a scan of a patient who was recently treated at Radboud University Medical Center. The corresponding geometries are in fig. 3.2 and, e.g., in fig. 3.30(b). These are both side-wall aneurysm geometries that can be well represented in the IB masking function. We could successfully simulate the flow in such geometries, provided the spatial resolution was adequate. We used typically 2310000

grid cells to cover the entire flow domain. An example of the steady flow obtained in Nijmegen-1 is shown in fig. 7.1.

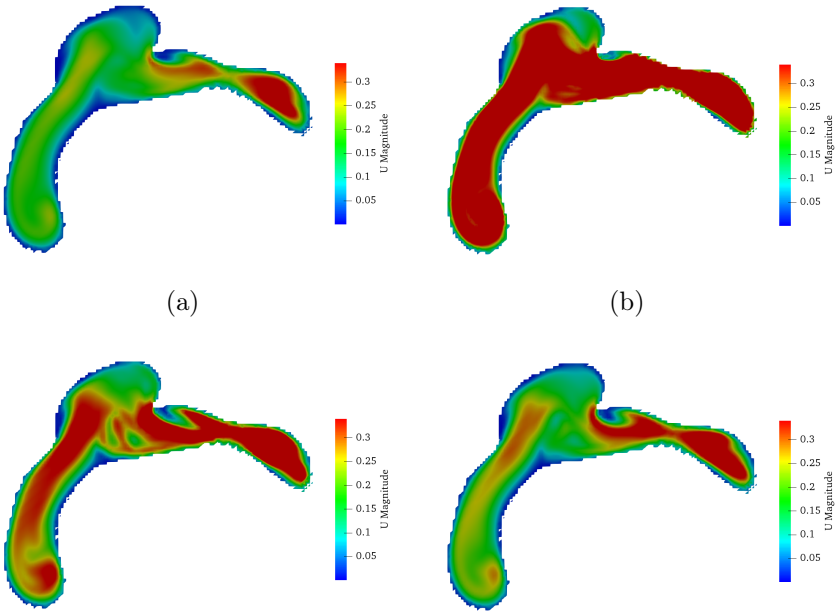


(a) (b)

Figure 7.1: A general impression of the steady flow in Nijmegen-1 geometry (a) in actual scale (b) in log scale.

- The second problem that was addressed is that of *reliably simulating time-dependent pulsatile flow in realistic, un-stented geometries*. This was illustrated on the basis of the Portugal geometry. Provided the time-step is sufficiently small, typically such that the Courant number is below 0.5 throughout the entire simulation, a precise prediction of the flow structures is achievable. Particular attention should be given to the initial condition. We observed that simulations that start 'impulsively' from rest can become unstable while starting from a non-zero steady state as was computed earlier, could yield a successful completion of a simulation. The method of 'continuation' may therefore be helpful to achieve simulations of relevant flow conditions, starting from an initial condition of a nearby situation. Some snapshots of the flow inside the aneurysm sac of the Portugal case are shown in fig. 7.2. The continuation approach should be considered in more detail to allow an easy traversal of a large section of the flow parameter space and treat the flow problem in various physiological conditions, e.g., 'rest' and 'active'.

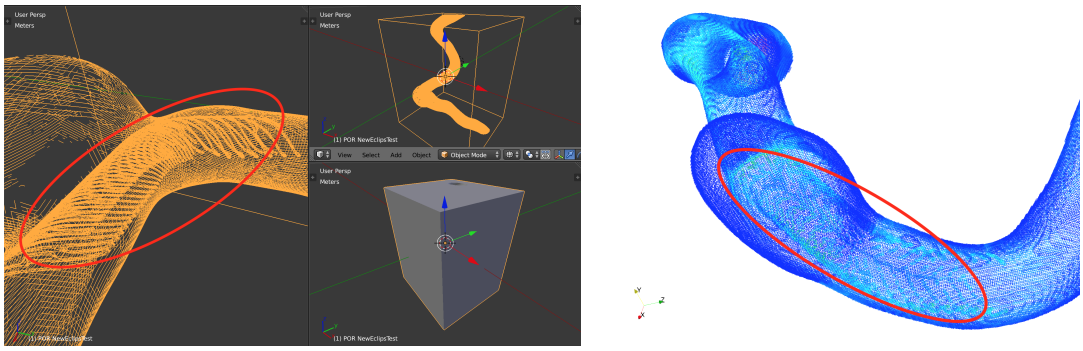
- The third problem, specific to simulation of flow in stented aneurysms is that of incorporating the model of the stent in the available realistic vessel geometry. Depending on the sizes of the inflow/outflow vessel connecting to the vessel section containing the aneurysm sac the generic shape of the stent needs to be scaled locally to fit the diameters. This is currently a task with many manual steps and somewhat heuristic interpretations on behalf of the operator, on how the stent should be deployed. This is a limiting factor at the moment and a topic that deserves a more systematic approach resting on optimization and structural mechanics input. An example of an implanted eclips device in the Portugal geometry is shown in fig. 7.3.



(a) (b)

(c) (d)

Figure 7.2: Snapshots of the flow inside the aneurysm sac of the Portugal geometry at (a) peak systole, (b) end systole, (c) peak diastole, and (d) end diastole.



(a) (b)

Figure 7.3: eCLIPs intergrated with Portugal geometry as seen in (a) Blender, and (b) ParaView.

To allow a faster turn-around time from DICOM data to flow analysis, some of the steps in the work-flow could be mapped onto typical shapes and connection topologies to allow some more automization. Also, the inclusion of the stents needs to be developed further, bringing practical experience from the clinic into an algorithmic stent placement. Also, the simulation times for realistic flow-diverting stents could become rather high - speed-up of the simulations is an additional, very important pacing item for future research.

Bibliography

[1] Horos project: About. URL https://horosproject.org/about/. Last visited on 11 May 2018.

[2] User guide. URL https://www.openfoam.com/documentation/user-guide/. Last visited on 19 December 2017.

[3] Eclips features, 2017. URL https://www.evasc.com/features. Last visited on 6 April 2018.

[4] Blender: About, 2018. URL https://www.blender.org/about/. Last visited on 19 June 2018.

[5] What are my treatment options?, 2018. URL http://brainaneurysm.com/what-are-my-treatment-options/. Last visited on 6 April 2018.

[6] F. Briganti, G. Leone, M. Marseglia, G. Mariniello, F. Caranci, A. Brunetti, and F. Maiuri. Endovascular treatment of cerebral aneurysms using flow-diverter devices: A systematic review. *The Neuroradiology Journal*, 28(4):365–375, 2015. doi: 10.1177/1971400915602803.

[7] R. D. Brown Jr and J. P. Broderick. Unruptured intracranial aneurysms: epidemiology, natural history, management options, and familial screening. *The Lancet Neurology*, 13:393–404, April 2014. doi: 10.1016/S1474-4422(14)70015-8.

[8] G. Chen, Q. Xiong, P. J. Morris, E. G. Paterson, A. Sergeev, and Y.-C. Wang. Openfoam for computational fluid dynamics. *Notices of the AMS*, 61(4):354–363, April 2014. doi: http://dx.doi.org/10.1090/noti1095.

[9] A. H. Chiu, J. de Vries, C. J. O'Kelly, H. Riina, I. McDougall, J. Tippett, M. Wan, A. L. de Oliveira Manoel, and T. R. Marotta. The second-generation eclips endovascular clip system: Initial experience. *Journal of Neurosurgery*, 128:482–489, January 2018. doi: 10.3171/2016.10.JNS161731.

[10] E. Constant, J. Favier, M. Meldi, P. Meliga, and E. Serre. An immersed boundary method in openfoam: verification and validation. *Computers and Fluids*, 2017. doi: 10.1016/j.compfluid.2017.08.001.

[11] A. Gambaruto, J. Janela, A. Moura, and A. Sequeira. Sensitivity of hemodynamics in a patient specific cerebral aneurysm to vascular geometry and blood rheology. *Mathematical Biosciences and Engineering*, 8(2):409–423, April 2011. doi: 10.3934/mbe.2011.8.409.

[12] F. P. Kärrholm. *Numerical Modelling of Diesel Spray Injection, Turbulence Interaction and Combustion*. PhD thesis, Chalmers University of Technology, Göteborg, 2008.

[13] M. T. Lawton and G. E. Vates. Subarachnoid haemorrhage. *The New England Journal of Medicine*, 377:257–266, July 2017. doi: 10.1056/NEJMcp1605827.

[14] T. R. Marotta, H. A. Riina, I. McDougall, D. R. Ricci, and M. Killer-Oberpfalzer. Physiological remodeling of bifurcation aneurysms: preclinical results of the eclips device. *Journal of Neurosurgery*, 128:475–481, February 2018. doi: 10.3171/2016.10.JNS162024.

[15] J. Mikhal. *Modeling and Simulation of Flow in Cerebral Aneurysms*. PhD thesis, University of Twente, Enschede, 2012.

[16] J. Mikhal and B. J. Geurts. Immersed boundary method for pulsatile transitional flow in realistic cerebral aneurysms. *Computers and Fluids*, pages 144–163, 2014. doi: 10.1016/j.compfluid.2013.12.009.

[17] J. Mikhal, G. M. Ong, J. de Vries, G. de Jong, R. Aquarius, J. Boogaarts, and B. J. Geurts. Toward automated analysis of flow in stented aneurysms. In *Proceedings CMFF '18*, 2018.

[18] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005. doi: 10.1146/annurev.fluid.37.061903.175743.

[19] A. Narata, F. de Moura, I. Larrabide, C. Perrault, F. Patat, R. Bibi, S. Velasco, A.-C. Januel, C. Cognard, R. Chapot, A. Bouakaz, C. Sennoga, and A. Marzo. The role of hemodynamics in intracranial bifurcation arteries after aneurysm treatment with flow-diverter stents. *American Journal of Neuroradiology*, 39:323–330, February 2018. doi: 10.3174/ajnr.A5471.

[20] H. Nilsson. A look inside icofoam (and pisofoam), 2017. URL https://pingpong.chalmers.se/public/pp/public_courses/course08331/published/1519801239304/resourceId/4037559/content/UploadedResources/aLookInsideIcoFoam-1.pdf. Last visited on 7 August 2018.

[21] G. M. Ong. *Internship report: Modeling flow in stented aneurysm*. Radboud UMC and University of Twente, April 2018.

[22] V. M. Pereira, O. Brina, B. M. A. Delattre, R. Ouared, P. Bouillot, G. Erceg, K. Schaller, K.-O. Lovblad, and M.-I. Vargas. Assessment of intra-aneurysmal flow modification after flow diverter stent placement with four-dimensional flow mri: a feasibility study. *Journal of NeuroInterventional Surgery*, 7:913–919, 2015. doi: 10.1136/neurintsurg-2014-011348.

[23] C. Poelma, P. N. Watton, and Y. Ventikos. Transitional flow in aneurysms and the computation of haemodynamic parameters. *Journal of the Royal Society Interface*, 12(105), 2015. doi: 10.1098/rsif.2014.1394.

[24] J. Xiang, R. J. Damiano, N. Lin, K. V. Snyder, A. H. Siddiqui, E. I. Levy, and H. Meng. High-fidelity virtual stenting: modeling of flow diverter deployment for hemodynamic characterization of complex intracranial aneurysms. *Journal of Neurosurgery*, 123:832–840, October 2015. doi: 10.3171/2014.11.JNS14497.